Complexity of Certificates, Heuristics, and Counting Types, with Applications to Cryptography and Circuit Theory

Habilitationsschrift

vorgelegt am 3. Februar 1999

der Fakultät für Mathematik und Informatik
der Friedrich-Schiller-Universität Jena

von
Dr. Jörg-Matthias Rothe

aus Erfurt

Gutachter:

1. Prof. Dr. Gerd Wechsung
2. Prof. Dr. Lane A. Hemaspaandra
3. Prof. Dr. Klaus W. Wagner

Erteilung der Lehrbefähigung am 21. Juni 1999.

To Irene and Paula,

my wife and my daughter

Acknowledgments

I wish to express my deep indebtedness to Professor Gerd Wechsung for his kind guidance and insightful advice, for his generosity, and for his constant and warm encouragement and support over the past decade. Working with him and learning from him is a very special pleasure. His scientific élan, his style, his inimitable enthusiasm have inspired generations of his students, both in mathematics and in computer science, and I consider it a privilege to be one of them. In particular, I am very grateful to him for bringing about my collaboration with Professor Lane A. Hemaspaandra.

Words cannot fully acknowledge the great debt I owe to Professor Lane A. Hemaspaandra. Working with him and learning from him is a unique experience. The present thesis as well as our many joint papers have greatly benefitted from his creative energy and from his careful way of turning ideas into formal concepts and correct proofs. In particular, I am deeply indebted to him for allowing me to take part in, and to contribute to, many of his research projects. Without this seminal joint work, the present thesis had not been possible.

Much of the research described in this thesis was done while I was with Professor Lane A. Hemaspaandra at the Computer Science Department of the University of Rochester as a postdoctoral research fellow. I thank him and his wife, Professor Edith Hemaspaandra, for many seminal and incisive discussions, for their warm hospitality, and in particular for introducing me to the delicacy of sushi. I very much enjoyed working with both of them.

The present thesis contains material from various research papers some of which originate from joint work with fellow researchers. Chapter 3 is based on joint work with Lane A. Hemaspaandra and Gerd Wechsung that appeared in journal as [HRW97a] and in conference proceedings as [HRW97b]. Chapter 4 presents joint work with Lane A. Hemaspaandra, where the results of Section 4.2 appeared in journal as [HR99], and the results of Section 4.3 are from [RH96], see also [HRW97b]. Section 5.3 describes joint research with Edith Hemaspaandra [HR98a], see also [HHR97c]. Chapter 6 presents joint work with Lane A. Hemaspaandra that is to appear in journal as [HR] and was published in conference proceedings as [HR98b]. Chapter 8 is based on joint work with Judy Goldsmith and Mitsunori Ogihara [GOR] that appeared in conference proceedings as [GOR98]. I thank all my coauthors for their generous permission to present the results of our joint work in

this thesis.

Further, I thank the following individuals for many interesting discussions, useful comments, and helpful hints to the literature: Eric Allender, Christer Berg, Bernd Borchert, Lance Fortnow, Judy Goldsmith, Erich Grädel, Edith Hemaspaandra, Lane Hemaspaandra, Gabriel Istrate, Andrew Klapper, Johannes Köbler, Dieter Kratsch, Mitsunori Ogihara, S. Ravi, Kenneth Regan, Alan Selman, Richard Stearns, Dietrich Stoyan, Staffan Ulfberg, Heribert Vollmer, Klaus Wagner, Gerd Wechsung, and roughly a dozen of anonymous[1] journal and conference referees of the various research papers presented in this thesis.

I thank Professor Gerd Wechsung, Professor Lane A. Hemaspaandra, and Professor Klaus W. Wagner for kindly agreeing to serve as expert referees for this thesis.

The Deutscher Akademischer Austauschdienst (DAAD) generously supported in part my research at the University of Rochester by a Postdoctoral Science Fellowship in the NATO program, and also by grants NSF-INT-9513368/DAAD-315-PRO-fo-ab and NSF-INT-9815095/DAAD-315-PPP-gü-ab. The National Science Foundation (NSF) of the USA supported in part my research under grant NSF-CCR-9322513. Part of this research was done while I was visiting the Department of Computer Science of the University of Rochester, the Mathematics Department of Le Moyne College, Syracuse, NY, and the Department of Computer Science of the University of Kentucky, and I thank the host institutions for their kind hospitality. I thank Lane A. Hemaspaandra for his financial support that allowed me to present the results of Chapter 7 (see [Rot, Rot98b]) at a conference in Winnipeg, Ontario.

Furthermore, I thank my colleagues in Jena: Harald Hempel, Maren Hinrichs, Dieter Kratsch, Gerhard Lischke, Haiko Müller, and Jörg Vogel.

The deepest gratefulness I owe to my wife, Irene, for her helpful advice and understanding, for our wonderful time in Rochester and Jena, and for her love. I thank Paula, our beloved daughter, for providing a most creative working environment during the months I was staying home with her and writing up this thesis. I also thank Paula for eagerly helping every day to tidy this mess up before Irene comes home from work. Last but not least, I thank my parents and all our friends in Rochester and Jena.

[1] In case you are upset because your name is missing on this list, just send an email to `rothe@informatik.uni-jena.de` and identify yourself.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computational complexity theory has developed into a rich, established field of theoretical computer science over the past three decades. It has by now produced a vast number of important results many of which fascinate both by their pure mathematical beauty and by their ability to stimulate and interact with other, in many cases applied, fields of computer science. To name a few examples, areas as diverse as cryptography, coding and information theory, data compression, logic, graph theory, the design and analysis of algorithms, or circuit theory have to date greatly benefitted from—and have had themselves a great deal of influence on—complexity theory.[1] Even in areas seemingly as far apart as political science and social choice theory, complexity-theoretic techniques were recently applied with striking success [BTT89a, BTT89b, BTT92, HHR97a, HHR97b]. Historically and conceptually, complexity theory is closely related to recursive function theory and to the theory of automata and formal languages. Finally, the young fields of quantum computing [Ber97] and biological computing [KMRS97]—which may have the potential to have a decisive impact on future computer technologies—dwell well and flourish in complexity theory.

One central task of complexity theory is to classify problems—that arise naturally in a wide variety of fields—with respect to their intrinsic computational complexity, i.e., to place them into complexity classes. Any complexity class contains all those problems that are solvable via a certain type of algorithm—specifically, via a Turing machine [Tur36] representing a certain computational paradigm such as deterministic computation, nondeterministic computation, probabilistic computation, etc.—and subject to a certain limitation of computational resources such as computation time or space.[2] By exploring the relation-

[1] Some of those areas are now so deeply interwoven with complexity theory that it is sometimes difficult to draw a firm borderline between them. Occasionally, this interaction has even created new, intermediate subfields such as complexity-theoretic cryptography or circuit complexity theory.

[2] Time (i.e., the number of steps the Turing machines takes to solve the problem), space (i.e., the number of memory cells needed), and other resources are measured in terms of functions of the input size. Throughout

ships between complexity classes, complexity theoreticians seek to learn the relative power of the underlying computational paradigms and/or resource constraints. However, the purpose of complexity theory goes beyond that. By studying the structure and properties of complexity classes, complexity theoreticians seek to gain insights into *why* some problems seem to be computationally harder than others, not merely to determine *how* easy or hard a given problem is to solve. This task is especially crucial in light of the fact that for many practically important problems no efficient algorithms appear to exist.

The most fundamental complexity classes are P (deterministic polynomial time) and NP (nondeterministic polynomial time). P was perceived as early as in the mid sixties [Cob64, Edm65] to be the most appropriate formal notion to capture the informal term of "feasible" computation: Sets contained in P are viewed as computationally easy sets. On the other hand, the hardest sets in NP [Coo71, Lev73] are considered to be computationally intractable. From the beginnings of complexity theory, the famous P $\stackrel{?}{=}$ NP question has been its central issue and its most serious challenge, if not the central issue and challenge of all of theoretical computer science, and it is an open question still today. The outstanding role of this question for both theory and practice is substantiated by the annoying discrepancy between, on the one hand, the utmost practical significance of many of the currently known NP-complete problems and, on the other hand, the total lack of efficient algorithms for any of them, despite the continuous, decades-long effort to get hold of such algorithms.

The present thesis continues the study of the structure and properties of the classes P and NP, and of related classes. More to the point, the main topics this thesis covers may be summarized by the following catchwords, which will be explained in more detail in the sequel: *certificate complexity*, *one-way functions*, *heuristics versus* NP*-completeness*, and *counting complexity*, where the latter item has three independent facets that may be described more specifically as: (a) *counting properties of circuits*, (b) *separations with immunity for counting classes*, and (c) *counting the solutions of tally* NP *sets*.

We will use the following concrete example of a problem in NP to describe more vividly some of the above topics. First, we need the following graph-theoretic notion. Given a graph G, a subset I of the vertex set of G is an *independent set* of G if no two vertices of I are joined by an edge. Let *mis*(G) denote the size (i.e., the number of vertices) of a maximum independent set of G. Figure 1.1 shows a graph G having two maximum independent sets each of size seven, i.e, *mis*$(G) = 7$. The problem `Independent Set` is defined as follows: Given a pair $\langle G, k\rangle$, where G is a graph and k is a positive integer, is it true that

this thesis, we consider only the traditional model of *worst-case complexity* (as opposed to average-case complexity, an alternative model that has also attracted much attention); i.e., resource functions $r(n)$ are bounds on the *maximum* resource allowed, where the maximum is taken over all inputs of size n.

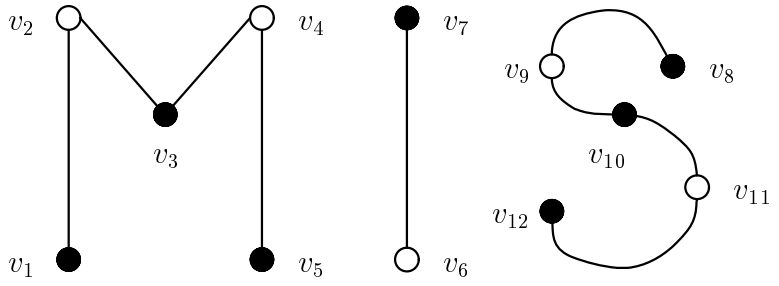

Figure 1.1: Graph G with one of its maximum independent sets displayed by full circles

mis$(G) \geq k$? `Independent Set` is one of the standard NP-complete problems,[3] i.e., it is one of the hardest problems in this class, since *every* NP problem can be transformed into (in technical terms: can be many-one reduced to) `Independent Set` in polynomial time.

Now we turn to the detailed description of the topics treated in the present thesis. We also provide some motivation and background of the questions to be addressed and the concepts to be used, and we present and discuss our main results as well as a number of related results known from the literature. For each topic discussed, we here give a detailed account of the publications and papers that originally describe the research presented in this thesis—for the publications overview, the reader is referred to the Acknowledgments.

(1) Certificate complexity – [HRW97a, HRW97b]. Given an NP set L and an instance $x \in L$, a *certificate* (a.k.a. *witness* or *solution*) for "$x \in L$" is a short string (i.e., a string of length at most polynomial in the length of x) that certifies membership of x in L so that this can be checked deterministically in polynomial time. For example, if G is the graph displayed in Figure 1.1, then $\langle G, 7\rangle \in$ `Independent Set`, and the two certificates for this membership are the vertex sets $\{v_1, v_3, v_5, v_6, v_8, v_{10}, v_{12}\}$ and $\{v_1, v_3, v_5, v_7, v_8, v_{10}, v_{12}\}$, appropriately encoded as strings. To check the validity of such a certificate, one simply has to verify that it contains seven vertices no two of which are joined by an edge, which is easy to do given the adjacency list of graph G.

Technically speaking, for any certificate scheme (i.e., NP machine) N accepting L, each accepting computation path of N on input x is a certificate for "$x \in L$." A certificate scheme N for L is said to be *easy* if there is a polynomial-time computable function that prints, for each $x \in L$, some certificate witnessing membership of x in L.

[3] Thousands of problems have as yet been shown to be NP-complete; see the book by Garey and Johnson [GJ79] for an early, yet very instructive account. However, due to the well-known fact that if any one NP-complete set is in P then all NP sets are in P, it does not really matter which particular NP-complete problem we pick as an example. Indeed, the famous Isomorphism Conjecture of Berman and Hartmanis [BH77] suggests that *all* NP-complete problems may, in fact, be just one and the same problem "in disguise."

Can easy sets only have easy certificate schemes? In Chapter 3, we study the class of sets satisfying that *all* their certificate schemes are easy. We denote this class by $\mathrm{EASY}^{\forall}_{\forall}$. We also study $\mathrm{EASY}^{\exists}_{\forall}$, the class of sets for which there exists *some* easy certificate scheme; it turns out that $\mathrm{EASY}^{\exists}_{\forall} = \mathrm{P}$. In addition, we study $\mathrm{EASY}^{\forall}_{\mathrm{io}}$ and $\mathrm{EASY}^{\exists}_{\mathrm{io}}$, the analogs of $\mathrm{EASY}^{\forall}_{\forall}$ and $\mathrm{EASY}^{\exists}_{\forall}$ whose certificate schemes are only required to be easy infinitely often. See Figure 3.1 on page 22 for the inclusions among these four classes within NP.

The idea captured by $\mathrm{EASY}^{\forall}_{\forall}$ was previously studied by Borodin and Demers [BD76]. Stating their result in our notation, they proved that if $\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP}$ then $\mathrm{P} \not\subseteq \mathrm{EASY}^{\forall}_{\forall}$, where coNP is the class of sets whose complements are in NP. That is, under a very plausible hypothesis it holds that *there exists an easy set that does not have only easy certificate schemes*. This beautiful result motivated the research reported on in Chapter 3.

Section 3.2 characterizes the classes $\mathrm{EASY}^{\forall}_{\forall}$ and $\mathrm{EASY}^{\forall}_{\mathrm{io}}$ in terms of Kolmogorov complexity, showing that they are robust. Informally speaking, Kolmogorov complexity ([Kol65, Cha66], see also [LV93]) measures the "randomness" of finite binary strings in an information-theoretic sense, and is here invoked to describe the *randomness of certificates*. That is, a certificate being "random" means it is "hard to find." In contrast, easy certificate schemes have certificates of small generalized Kolmogorov complexity.[4]

In our example, if some certificate for "$\langle G, 7\rangle \in$ `Independent Set`" happens to be encoded as the binary string consisting of only zeroes, this certificate would not be Kolmogorov-random. However, NP-complete problems such as `Independent Set` are very unlikely to have easy certificate schemes, since due to $\mathrm{EASY}^{\exists}_{\forall} = \mathrm{P}$ this property would immediately imply $\mathrm{P} = \mathrm{NP}$. Interestingly, NP-complete sets have *some* easy certificate scheme if and only if they have *only* easy certificate schemes, see Corollary 3.3.2 on page 28. In fact, the difficulty of *finding* certificates for the accepted instances of NP-complete problems appears to be one crucial source of their intractability. The point of the result of Borodin and Demers and of our results is that even *easy sets* (i.e., sets in P) are likely to have *hard certificate schemes* (i.e., certificate schemes not always having Kolmogorov-easy certificates).

Section 3.3 provides structural conditions—regarding immunity,[5] P-printability,[6] and class collapses—that put upper and lower bounds on the sizes of the classes $\mathrm{EASY}^{\forall}_{\forall}$ and $\mathrm{EASY}^{\forall}_{\mathrm{io}}$. The positive results of Section 3.3 are summarized in Figure 3.2 on page 26.

[4] Note that "randomness" means "non-compressibility" in terms of the theory of data compression. Generalized Kolmogorov complexity was introduced by Hartmanis [Har83a] to measure not only how far a string can be compressed, but also how fast it can be "restored." See Definition 3.2.5 on page 23.

[5] Immunity is an important notion of both recursive function theory [Rog67] and complexity theory. For any complexity class $\mathcal{C}$, a *$\mathcal{C}$-immune* set is an infinite set having no infinite subset in $\mathcal{C}$.

[6] P-printability [HY84] is a notion arising in the theory of Kolmogorov complexity and data compression. Informally stated, a set is P-printable if all its elements up to a given length can be printed in polynomial time, see Definition 2.2.8 on page 17.

Section 3.4 provides negative results showing that some of our positive claims are optimal with regard to being relativizable. One such result is proven via a novel observation: the classical "wide spacing" oracle construction technique yields instant non-bi-immunity results. We also construct an oracle relative to which $\mathrm{NP} \neq \mathrm{P} = \mathrm{EASY}^{\forall}_{\forall}$, which improves on the classical result that, in some relativized world, $\mathrm{NP} \neq \mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$ [BGS75].

(2) One-way functions. The previous comments on NP-complete problems lacking efficient algorithms, yet being of extreme practical importance, might have given the (wrong) impression that computational efficiency was the only criterion of deciding what is useful, important, desirable for practical purposes. On the contrary, computational *in*efficiency (hardness)—and above all, *provable* hardness—is sometimes useful and desirable in certain applications. In particular, applications having to do with strategic considerations, with adversaries expected to have a certain computational power, with security, etc. undoubtedly do need, and rely upon, computational hardness and do need, and rely upon, theoretically well-founded results proving hardness to be an inherent feature of the given computational task. One such field instantly comes to mind: cryptography.[7]

In cryptography, and particularly in its modern subfields such as public-key cryptography, one seeks to find secure ways of transmitting data in ciphered form over unsecure channels, thus seeking to guarantee that the information transmitted is secure against eavesdropping, forgery of authorship, etc. To this end, appropriate cryptographic protocols need be designed whose building blocks are cryptographic primitives such as pseudo-random generators, bit commitment schemes, or one-way functions. Intuitively, a one-way function is a function that is easy to compute but hard to invert. Since the present thesis is thematically concerned with complexity theory, we will be dealing with complexity-theoretic cryptography. In particular, the type of one-way functions we will study in Chapter 4 is the complexity-theoretic one-way functions introduced by Grollmann and Selman [GS88].

An issue of central importance in complexity-theoretic cryptography is the question of whether or not such functions exist. However, since it is well known that one-way functions exist if and only if $\mathrm{P} \neq \mathrm{NP}$, we cannot give an ultimate answer to this question unless we can solve the $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ question. All we can hope for is to characterize the existence of certain special types of one-way functions in terms of complexity-theoretic conditions such as complexity class separations. Chapter 4 provides such characterizations for certain types of associative one-way functions—which were introduced by Rabi and Sherman [RS97]—and for partial and total one-way permutations. A one-way permutation is an injective (i.e., one-to-one) and surjective (i.e., onto) one-way function.

[7] This field, sure enough, is not the only one. For instance, a very interesting application field of complexity-theoretic hardness results in political science was recently proposed by Bartholdi et al. [BTT89a]: They prove that for the Copeland voting scheme, an election scheme that is in practical use today, the computational task of manipulating the election is hard; in fact, is NP-complete.

(2a) Associative one-way functions – [HR99]. Rabi and Sherman [RS97] presented novel digital signature and unauthenticated secret-key agreement protocols, developed by themselves and by Rivest and Sherman. These protocols use strong,[8] total, commutative (in the case of multi-party secret-key agreement), associative one-way functions as their key building blocks. Though Rabi and Sherman did prove that associative one-way functions exist if P $\neq$ NP, they left as an open question whether any natural complexity-theoretic assumption is sufficient to ensure the existence of strong, total, commutative, associative one-way functions. In Section 4.2, we prove that if P $\neq$ NP then strong, total, commutative, associative one-way functions exist.

The proof of this result also resolves another question of Rabi and Sherman. They also posed as an open issue the problem of whether a strong, total, associative one-way function can be constructed from any given one-way function [RS93]. The proof of Theorem 4.2.8 shows how to create a strong, total, commutative, associative one-way function from any set in NP $-$ P, assuming P $\neq$ NP. For example, if P $\neq$ NP then `Independent Set` $\in$ NP $-$ P, and thus our proof shows how to construct such a one-way function from this particular problem. Since it is well known how to construct a set in NP $-$ P from any given one-way function, Rabi and Sherman's question can be answered in the affirmative.

Furthermore, based on Kleene's [Kle52] careful distinction between *weak* and *complete equality* of partial functions, we give a novel definition of associativity that, for partial functions, is a more natural analog of the standard total-function definition than that of Rabi and Sherman. We show that their and our results hold even under this definition.

We also discuss, in Section 4.2, the issue of injective associative one-way functions. Though Rabi and Sherman proved that no total associative one-way function can be injective, we characterize the existence of partial, injective, associative one-way functions by the separation P $\neq$ UP, where UP [Val76] (unambiguous polynomial time) is the class of those sets L in NP that have some certificate scheme for which each $x \in L$ has exactly one certificate. UP has long played a central role in complexity-theoretic cryptography; see, e.g., [GS88]. By definition, P $\subseteq$ UP $\subseteq$ NP, and both inclusions are widely suspected to be proper—though this is an open research problem: any proof of P $\neq$ UP or UP $\neq$ NP would immediately prove that P $\neq$ NP. NP-complete problems such as `Independent Set` are unlikely to be in UP.[9] Candidates for problems in UP $-$ P are, e.g., (a language version of) the discrete logarithm problem [GS88] and primality testing [FK92].

[8] Informally stated, strong one-way functions are binary one-way functions that are hard to invert even if one of their arguments is given, see Definition 4.2.7 on page 49.

[9] See the papers [HH88a, Rac82] for arguments as to why. Intuitively, instances $\langle G, k\rangle$ of `Independent Set` may have up to exponentially many (in the size of $\langle G, k\rangle$) certificates in the canonical certificate scheme (i.e., for the "natural" NP algorithm that guesses all possible subsets of size $\geq k$ of the vertex set and, for each subset guessed, checks whether or not it is an independent set), and it seems unlikely that `Independent Set` has some other certificate scheme obeying the very restrictive unambiguity requirement of UP.

Finally, we provide a counterexample to a construction that Rabi and Sherman claim converts partial associative one-way functions into total associative one-way functions.[10] More precisely, we show that any proof of their claim would immediately prove UP $=$ NP.

(2b) Partial and total one-way permutations – [RH96], see also [HRW97b]. A one-way permutation is an injective and surjective one-way function, and in Section 4.3 we consider both partial and total one-way permutations. Interestingly, it turns out that partial one-way permutations are closely linked with the easy certificate classes introduced in Chapter 3. In particular, we define the UP analog of $\mathrm{EASY}^{\forall}_{\forall}$, denoted $\mathrm{EASY}^{\forall}_{\forall}(\mathrm{UP})$. This class *simultaneously* prevents two of the possible sources of the potential intractability of NP-completeness: It reduces the solution space of NP problems to at most one solution for each input, and it requires that this one solution, if it exists, is easy to find. Expanding results of Grollmann and Selman [GS88], we show that the existence of partial one-way permutations can be characterized by the separation $\mathrm{P} \neq \mathrm{EASY}^{\forall}_{\forall}(\mathrm{UP})$.

We also characterize the existence of various types of poly-to-one one-way functions in terms of separations such as $\mathrm{P} \neq \mathrm{EASY}^{\forall}_{\forall}(\mathrm{FewP})$, where $\mathrm{EASY}^{\forall}_{\forall}(\mathrm{FewP})$ is the FewP analog of $\mathrm{EASY}^{\forall}_{\forall}$. FewP [All86, AR88] (ambiguity-bounded polynomial time) is that subclass of NP whose machines have at most polynomially many (in the input size) certificates for each input. Finally, we establish a condition necessary and sufficient for the existence of total one-way permutations.

(3) Heuristics versus NP-completeness. Theoretical hardness results—such as NP-completeness results—are often thought of as being negative results: They express the impossibility to efficiently solve important problems. A practitioner who urgently needs to have some problem solved might say: "I don't care about theory. All I care about are those 2500 instances of `Independent Set` that I must have solved by tomorrow morning. And as far as I'm concerned, for my first 423 inputs this greedy heuristic seems to have worked just perfectly."

Indeed, heuristic algorithms can be useful in quickly providing solutions to certain "well-structured" instances of hard problems. For instance, our example graph G in Figure 1.1 happens to have only vertices of degree 1 or 2, and it is known that the `Independent Set` problem, restricted to graphs with no vertex degree exceeding 2, is solvable in deterministic polynomial time, see [GJ79]. Similarly, the Minimum Degree Greedy Algorithm (MDG, for short; see Figure 5.1 on page 68) efficiently finds a maximum independent set if the input graph is a tree, a split graph, the complement of a k-tree, a well-covered graph, or a complete k-partite graph. Even in cases where some heuristic fails to find the optimum,

[10] Their claim refers to the notion of associativity that is based on *weak* partial-function equality in Kleene's sense. Though their claim is invalid, we show that for associativity based on *complete* partial-function equality, a variant of Rabi and Sherman's construction indeed is useful in our proof of Theorem 4.2.8.

it still might be useful in providing good approximate solutions of the problem. Again, for certain graph classes it is known that MDG has a good approximation ratio [HR94].

The central question addressed in Chapter 5 is: *Given a hard problem, and given a heuristic algorithm for that problem, what is the complexity of recognizing on which inputs the heuristic does well?* To be a bit more specific about a heuristic "doing well," we mention the following three facets of this question, focusing on `Independent Set` and the MDG heuristic. First, one can restrict the *decision* problem `Independent Set` to those inputs on which MDG is able to solve it, and study the complexity of the restricted problem. Second, one can ask about the complexity of recognizing those graphs for which MDG can solve the *optimization* problem, i.e., for which MDG is able to output a maximum independent set. Third, given any fixed constant r, one can ask about the complexity of recognizing those graphs for which MDG can *approximate* the optimum solution within a factor of r.

We study this question for two important NP-complete graph problems: `Independent Set` and `K-Colorability`, see [GJ79]. Let $\chi(G)$ denote the chromatic number of graph G, i.e., the smallest number of colors needed to color the vertices of G such that no two adjacent vertices receive the same color. The problem `K-Colorability` is the following: Given a graph G and a positive integer k, is it true that $\chi(G) \leq k$? Note that already `3-Colorability`—the special case of `K-Colorability` with $k = 3$—is NP-complete.

(3a) Independent set problems – [HR98a], see also [HHR97c]. Bodlaender, Thilikos, and Yamazaki [BTY97] proposed to study the computational complexity of the problem of whether the MDG algorithm can approximate a maximum independent set of a given graph within a constant factor of r, for any fixed rational $r \geq 1$. (Note that the special case of $r = 1$ is the optimization problem alluded to above.) They denoted this problem by $\mathcal{S}_r$ and proved that for each rational $r \geq 1$, $\mathcal{S}_r$ is coNP-hard. They also provided a P^{NP} upper bound of $\mathcal{S}_r$, where P^{NP} is the class of sets solvable via sequential (a.k.a. "Turing" or "adaptive") access to NP. They left open the question of whether the gap between the upper and the lower bound of $\mathcal{S}_r$ can be closed. For the special case of $r = 1$, they showed that $\mathcal{S}_1$ is even DP-hard, where DP [PY84] is the class of sets that can be represented as the difference of two NP sets. Again, they left open the question of whether $\mathcal{S}_1$ can be shown to be complete for DP or some larger class such as P^{NP}.

In Section 5.3, we completely solve all the questions left open in [BTY97]. Our main result is that for each rational $r \geq 1$, $\mathcal{S}_r$ is complete for $\mathrm{P}_{||}^{\mathrm{NP}}$, the class of sets solvable via parallel (a.k.a. "truth-table" or "nonadaptive") access to NP. The class $\mathrm{P}_{||}^{\mathrm{NP}}$ has recently proven to be important for describing the complexity of some natural problems for which previously only NP-hardness or coNP-hardness lower bounds were known, see the papers [HHR97a, HHR97b, HW97] and the survey [HHR97c].

(3b) Graph coloring problems – [Rot98a]. In Section 5.4, we study the complexity

of the problem 3-Colorability when restricted to those input graphs on which a given graph coloring heuristic is able to solve the problem. The heuristics we consider include the sequential algorithm traversing the vertices of the graph in various orderings—e.g., in the order by decreasing degree or in the recursive smallest-last order of Matula et al. [MMI72]—as well as Wood's algorithm [Woo69]. For each of the seven heuristics considered in Section 5.4, we prove that the corresponding restriction of 3-Colorability remains NP-complete.

(4) Counting complexity. To precisely describe the complexity of computing the permanent of a given matrix, Valiant [Val79a] introduced the counting class #P. Counting has since been a central theme in complexity theory, and a rich body of very useful, important, sometimes surprising results has been developed; see, e.g., the papers [Hem87, Hem89, Tod91a, Tod91b, TO92, Tor88, Tor91] and the excellent surveys [Sch90, For97b]. Traditionally, NP is viewed as a class capturing the existential quantification: Does there *exist* a solution to a given instance? Valiant's paper and its follow-up papers ask about the *number* of solutions; i.e., #P is the class of functions that count the number of certificates of NP machines. For example, the canonical certificate scheme for Independent Set described in Footnote 9 corresponds to some $f \in \#\mathrm{P}$; taking, e.g., the graph G from Figure 1.1, we have $f(\langle G, 7\rangle) = 2$. #P functions can be used to define various important counting and probabilistic classes such as PP, BPP, $\mathrm{C_{=}P}$, $\oplus$P, and SPP; see Definition 2.2.1 on page 13. The study of #P and of related classes has significant applications in circuit complexity and other fields.

(4a) Counting properties of circuits – [HR, HR98b]. The mother of complexity theory is recursive function theory. One of the most beautiful and important results of recursive function theory is Rice's Theorem [Ric53, Ric56], which states that every nontrivial language property of the recursively enumerable sets is undecidable. Borchert and Stephan [BS97] initiated the search for complexity-theoretic analogs of Rice's Theorem. In particular, they proved that every nontrivial counting property of circuits is UP-hard,[11] and that a number of closely related problems are SPP-hard. SPP plays a central role in complexity theory [For97b] and, in particular, is closely linked to the closure properties of #P [OH93].

In Chapter 6, we show that Borchert and Stephan's UP-hardness result itself cannot be improved to SPP-hardness unless unlikely complexity class containments hold. Nonetheless, we prove that every "P-constructibly bi-infinite counting property of circuits"—see Definition 6.3.5 on page 93—is SPP-hard. We also raise their general lower bound from unambiguous nondeterminism to constant-ambiguity nondeterminism.

[11] See Definition 6.1.3 on page 86 for the precise meaning of those terms, and note in particular the nonstandard usage of "$\mathcal{C}$-hardness of counting properties of circuits" for any complexity class $\mathcal{C}$.

(4b) Separations with immunity for counting classes – [Rot, Rot98b]. Ko [Ko90] and Bruschi [Bru92] independently showed that, in some relativized world, PSPACE contains a set that is *immune* to PH, the polynomial hierarchy [MS72, Sto77]. A complexity class separation witnessed by an immune set is a particularly strong separation, since an immune witness set contains only finite sets of the class to separate from; immunity thus shields the witness set against being "approximated from the inside." In Chapter 7, we study and settle the question of relativized separations with immunity for PH and the counting classes PP, $\mathrm{G{=}P}$, and $\oplus$P in all possible pairwise combinations. Our results strengthen previously known simple separation results of Torán [Tor91], Green [Gre91], and Berg and Ulfberg [BU], which are not witnessed with immunity. We also prove the existence of a relativized $\mathrm{G{=}P}$-simple set (i.e., a coinfinite $\mathrm{G{=}P}$ set whose complement is $\mathrm{G{=}P}$-immune), which extends results of Balcázar et al. [Bal85, BR88]. Our proof technique requires an exponential circuit lower bound for the Boolean function $\mathrm{EQU}_n^{\mathrm{half}}$ (defined on page 100) that is derived from Razborov's [Raz87] circuit lower bound for the majority function.

(4c) Counting the solutions of tally NP sets – [GOR, GOR98]. Chapter 8 studies $\#\mathrm{P}_1$ [Val79b], the class of functions that count the number of solutions of *tally* NP sets. A set is tally if it is encoded over a unary alphabet. $\#\mathrm{P}_1$ is an interesting subclass of $\#$P and contains a number of important problems such as the problem `Self-Avoiding Walk` (defined on page 114, see also [Wel93]), a classical problem of statistical physics and polymer chemistry. The most significant question regarding $\#\mathrm{P}_1$ is whether or not $\#\mathrm{P}_1$ is contained in FP, the class of polynomial-time computable functions. We study this question in relation to other complexity-theoretic conditions. Note that if $\#\mathrm{P}_1 \subseteq$ FP then all tally NP sets are in P. We prove that the assumption $\#\mathrm{P}_1 \subseteq$ FP implies even more unlikely complexity class collapses: PH $\subseteq \oplus$P and P $=$ BPP.

We also show that $\#\mathrm{P}_1$ is contained in FP if and only if every P set has an easy (i.e., polynomial-time computable) census function. Informally stated, the census function of a set L maps each number n (given in unary) to the number of length n elements in L. Census functions are a central notion in complexity theory and have proven useful in many contexts, see Section 8.1. Our main result is that every $\#\mathrm{P}_1^{\mathrm{PH}}$ function can be computed in $\mathrm{FP}^{\#\mathrm{P}_1{}^{\#\mathrm{P}_1}}$. Consequently, every P set has an easy census function if and only if every set in the polynomial hierarchy does. We relate a set's property of having an easy census function to other well-studied properties of sets, such as P-printability, rankability [GS91] (another notion arising in the theory of Kolmogorov complexity and data compression), and scalability [GH96] (the closure of the rankable sets under P-isomorphisms).

Finally, we prove that it is no more likely that the census function of any set in P can be approximated (more precisely, can be n^α-enumerated in time n^β for fixed α and β) than that it can be precisely computed in polynomial time.

Chapter 2

Notations and Basic Concepts

This chapter provides the notations and the basic complexity-theoretic concepts used in this work. Readers familiar with the concepts and the standard notation of complexity theory are encouraged to skip the present chapter and to consult the Index when necessary. For further information and historical background, we refer to some standard text book on computational complexity such as [HU79, WW86, BDG88, BDG90, BC93, Pap94].

Further concepts and notations will be defined later on; e.g., the fundamental notion of Kolmogorov complexity will be defined in Section 3.2, and some basic graph-theoretic concepts will be provided in Section 5.2. The reader is assumed to be familiar with the standard notations of mathematics, logic, and computer science.

2.1 Strings, Sets, and Functions

Fix the two-letter alphabet $\Sigma = \{0, 1\}$. Σ^* is the set of all strings over Σ, and $\Sigma^+ = \Sigma^* - \{\epsilon\}$, where ϵ denotes the empty string. For each string $x \in \Sigma^*$, $|x|$ denotes the length of x. We denote the set of non-negative integers by $\mathbb{N}$, and we adopt the standard bijection between Σ^* and $\mathbb{N}$—the natural number i corresponds to the lexicographically $(i+1)$st string in Σ^*: $0 \leftrightarrow \epsilon$, $1 \leftrightarrow 0$, $2 \leftrightarrow 1$, $3 \leftrightarrow 00$, etc. We let $\leq$ denote the standard lexicographic order on Σ^*.

The existential quantifier is denoted by $\exists$, and the universal ("for all") quantifier is denoted by $\forall$. As is standard, the notation $\exists^{\text{io}}x$ (respectively, $\forall^{\text{ae}}x$) means "there exist infinitely many x" (respectively, "for all but finitely many x").

For each set $L \subseteq \Sigma^*$ (equivalently, $L \subseteq \mathbb{N}$ via the above bijection), $||L||$ denotes the cardinality of L, and $\overline{L} = \Sigma^* - L$ denotes the complement of L. For any class $\mathcal{C}$ of sets, define co$\mathcal{C} = \{L \mid \overline{L} \in \mathcal{C}\}$. Let $L^{=n}$ (respectively, $L^{\leq n}$) denote the set of strings in L of length n (respectively, of length at most n), and let $L^{<n} = L^{\leq n} - L^{=n}$. Let Σ^n and $\Sigma^{\leq n}$

be shorthands for $(\Sigma^*)^{=n}$ and $(\Sigma^*)^{\leq n}$, respectively. For any sets X and Y, their *disjoint union*, denoted $X \oplus Y$, is defined to be $\{0x \mid x \in X\} \cup \{1y \mid y \in Y\}$. For classes $\mathcal{C}$ and $\mathcal{D}$ of sets, let $\mathcal{C} \oplus \mathcal{D}$ denote $\{C \oplus D \mid C \in \mathcal{C} \text{ and } D \in \mathcal{D}\}$.

All functions may potentially be many-to-one and partial, unless explicitly specified to be one-to-one or total. For each (single-valued, partial or total) function f, let domain(f) and image(f) denote the domain and image of f, respectively.

To encode pairs of strings as a single string, we use some standard total, one-to-one, onto, polynomial-time computable pairing function, $\langle \cdot, \cdot \rangle : \Sigma^* \times \Sigma^* \to \Sigma^*$, that has polynomial-time computable inverses, and is non-decreasing in each argument when the other argument is fixed.[1] This pairing function is extended to encode m-tuples of strings as is standard. For convenience, we will sometimes write m-tuples of strings $x_1, x_2, \ldots, x_m \in \Sigma^*$ explicitly as $x_1 \# x_2 \# \cdots \# x_m$, using a special separating symbol $\#$ not in Σ.

The *characteristic function* of any set L is denoted by χ_L, i.e., $\chi_L(x) = 1$ if $x \in L$, and $\chi_L(x) = 0$ if $x \notin L$.

2.2 Turing Machines, Complexity Classes, Reducibilities, and Other Notions

The definition of Turing machines and their languages, Turing transducers and the functions they compute, relativized (i.e., oracle) computations, (relativized) complexity classes, etc. is standard in the literature, see, e.g., the text books [HU79, WW86, BDG88, BC93, Pap94]. We will abbreviate "deterministic (respectively, nondeterministic) polynomial-time Turing machine" by DPTM (respectively, NPTM). An *unambiguous* Turing machine (sometimes called *categorical* Turing machine) is a nondeterministic Turing machine that on no input has more than one accepting computation path.

For any Turing machine M, $L(M)$ denotes the *language* of M, i.e., the set of strings accepted by M. The notation $M(x)$ means "M on input x." For any NPTM N and any input x, we assume that all computation paths of $N(x)$ are suitably encoded by strings over Σ. An NPTM N is said to be *normalized* if there exists a polynomial q such that for all $n \in \mathbb{N}$, $q(n) \geq n$ and, on each input of length n, all computation paths of length $q(n)$ exist in the computation of $N(x)$, and $N(x)$ has only computation paths of length $q(n)$. Unless otherwise stated, all NPTMs considered in this work are required to be normalized.

We assume that the set of final states of any Turing machine is partitioned into the set of accepting states and the set of rejecting states. Computation paths of nondeterministic

[1] Using the above-mentioned standard correspondence between Σ^* and $\mathbb{N}$, we will view $\langle \cdot, \cdot \rangle$ also as a pairing function mapping $\mathbb{N} \times \mathbb{N}$ onto $\mathbb{N}$.

Turing machines are assumed to be encoded as strings in Σ^*; a computation path that terminates in an accepting final state is called an *accepting computation path*, and a computation path that terminates in a rejecting final state is called a *rejecting computation path*.

For any NPTM N and any input x, we denote the *set of accepting computation paths* of $N(x)$ by $\mathrm{ACC}_N(x)$, and we let $\mathrm{acc}_N(x) = ||\mathrm{ACC}_N(x)||$ denote the *number of accepting computation paths* of $N(x)$. Similarly, $\mathrm{rej}_N(x)$ denotes the *number of rejecting computation paths* of $N(x)$, and $\mathrm{tot}_N(x)$ denotes the *total number of computation paths* of $N(x)$.

For any oracle Turing machine M, we will use the same notations as given above, except that "M" be replaced by "M^A" when A is the given oracle set. In particular, the above normalization requirement is meant to hold independent of the specific oracle used. Sometimes, when we speak of an oracle Turing machine M with no oracle specified, we write $M^{(\cdot)}$. We will use the shorthands DPOTM and NPOTM, respectively, to denote DPTMs and NPTMs that are oracle Turing machines. We allow both languages and functions to be used as oracles. In the latter case, the model is the standard one, namely, when query q is asked to a function oracle f the answer is $f(q)$.

We briefly recall the definitions of some well-known complexity classes that are central to this work. In some of the chapters, a number of additional complexity classes will be defined. Every complexity class that occurs in this work can be found in the Index along with a reference to the page where it is defined. Figure 2.1 summarizes the known inclusions among (some of) the classes defined below.

P (respectively, NP) is the class of all sets that are accepted by some DPTM (respectively, NPTM). UP [Val76] is the class of all sets that are accepted by some unambiguous NPTM. PSPACE is the class of problems decidable in polynomial space. Let FINITE be the class of all finite sets. Let P_∞ denote the class $\mathrm{P} - \mathrm{FINITE}$ of all infinite P sets. FP denotes the class of all polynomial-time computable functions from Σ^* to Σ^*. FE is the class of functions that can be computed by deterministic transducers running in time 2^{cn} for some constant c. Let $\mathrm{E} = \bigcup_{c>0} \mathrm{DTIME}[2^{cn}]$ and $\mathrm{NE} = \bigcup_{c>0} \mathrm{NTIME}[2^{cn}]$.

Definition 2.2.1 *1.* [MS72, Sto77] *The polynomial hierarchy is inductively defined as follows:* $\Sigma_0^p = \mathrm{P}$, $\Sigma_k^p = \mathrm{NP}^{\Sigma_{k-1}^p}$ *and* $\Pi_k^p = \mathrm{co}\Sigma_k^p$ *for* $k \geq 1$, *and* $\mathrm{PH} = \bigcup_{i \geq 0} \Sigma_i^p$.

2. [Val79a, Val79b] $\#\mathrm{P} = \{\mathrm{acc}_M \mid M \text{ is an NPTM}\}$.

3. [Gil77] PP *is the class of languages* L *for which there exists an NPTM* M *such that for all strings* $x \in \Sigma^*$, $x \in L \iff \mathrm{acc}_M(x) \geq \mathrm{rej}_M(x)$.

4. [Sim75, Wag86] $\mathrm{C{=}P}$ *is the class of languages* L *for which there exists an NPTM* M *such that for all strings* $x \in \Sigma^*$, $x \in L \iff \mathrm{acc}_M(x) = \mathrm{rej}_M(x)$.

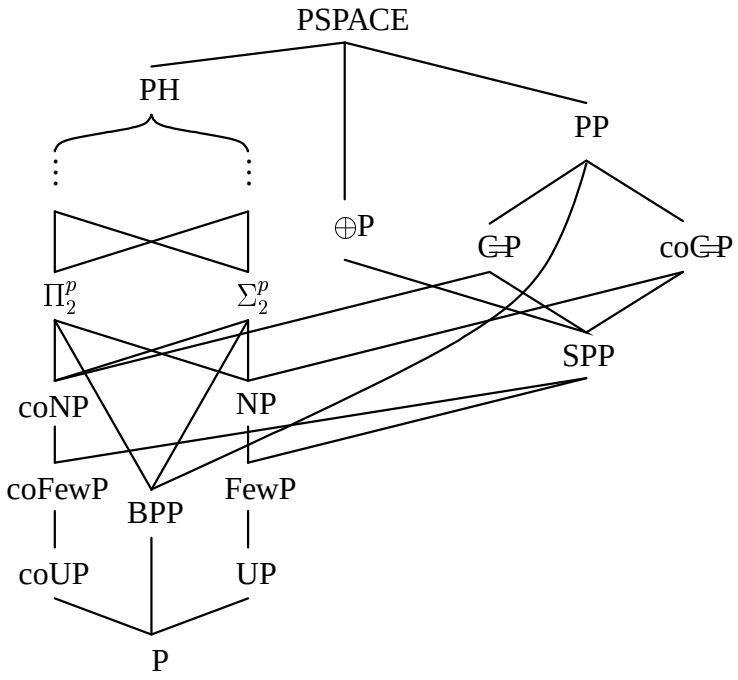

Figure 2.1: Inclusion relations among some complexity classes

5. [CH90, Her90, BG92] *For any fixed* $k \geq 2$, MOD_kP *is the class of languages* L *for which there exists an NPTM* M *such that for all strings* $x \in \Sigma^*$, $x \in L \iff \text{acc}_M(x) \not\equiv 0 \mod k$.

   If $k = 2$, *we write* $\oplus$P *(introduced in* [PZ83, GP86]*) instead of* MOD_2P.

6. [Gil77] BPP *is the class of languages* L *for which there exists an NPTM* M *such that for all strings* $x \in \Sigma^*$, $(x \in L \implies \text{rej}_M(x) \leq \frac{1}{4}\text{tot}_M(x))$, *and* $(x \notin L \implies \text{acc}_M(x) \leq \frac{1}{4}\text{tot}_M(x))$.

7. [All86, AR88] FewP *is the class of languages* L *for which there exist an NPTM* M *and a polynomial* p *such that for all strings* $x \in \Sigma^*$, (a) $\text{acc}_M(x) \leq p(|x|)$, *and* (b) $x \in L \iff \text{acc}_M(x) > 0$.

8. [OH93, FFK94] SPP *is the class of languages* L *for which there exist functions* $f \in \#\text{P}$ *and* $g \in \text{FP}$ *such that for all strings* $x \in \Sigma^*$, $(x \in L \implies f(x) = 2^{|g(x)|}+1)$, *and* $(x \notin L \implies f(x) = 2^{|g(x)|})$.

There are various equivalent ways of defining the classes presented in Definition 2.2.1 above. For instance, the levels of the polynomial hierarchy may alternatively be defined via

alternating polynomially bounded existential and universal quantifiers applied to a P predicate [Sto77, Wra77]. Counting classes such as PP, $\mathrm{C_{=}P}$, $\oplus$P, or SPP may be defined via the difference between the number of accepting and the number of rejecting computation paths of NPTMs [FFK94]—such classes are called "gap-definable classes" in [FFK94]. Bovet, Crescenzi, and Silvestri [BCS92] introduced the concept of "leaf languages" as an excellent way of *uniformly* defining virtually all standard complexity classes. Concluding this list of definitional alternatives, we mention that it has become popular among complexity theoreticians to define new language and function classes via applying operators to some given class of sets, and we will resort to this common operator notation at times. This very flexible formalism was especially successful with regard to yielding strong and unexpected results regarding the power of counting classes, as well as in the study of Wagner's counting hierarchy [Wag86]. The reader is referred to the work of Toda [Tod91a, Tod91b], Toda and Ogihara [TO92], Wagner [Wag86], Torán [Tor91], and Vollmer and Wagner [VW95, VW93], see also [Vol94].

We exemplify this operator-based approach by giving below the definition of the "#" operator,[2] which generalizes Valiant's class #P defined above and which may be elegantly used to yield generalizations of the above counting classes. We refer to Definition 6.2.1 on page 88 and to Definition 8.2.2 on page 116 for interesting refinements of the general # operator.

Definition 2.2.2 *For any language class $\mathcal{C}$, define $\# \cdot \mathcal{C}$ to be the class of functions $f : \Sigma^* \to \mathbb{N}$ for which there exist a set $A \in \mathcal{C}$ and a polynomial p such that for each $x \in \Sigma^*$, $f(x) = ||\{y \mid |y| = p(|x|)$ and $\langle x, y\rangle \in A\}||$.*

For any complexity class $\mathcal{C}$, we use the term $\mathcal{C}$ *machine* (respectively, $\mathcal{C}$ *oracle machine*) to denote a Turing machine (respectively, Turing oracle machine) whose specific acceptance/rejection behavior is implicitly given by the definition of $\mathcal{C}$. Given any (relativizable) complexity class $\mathcal{C}$ and any oracle set A, we write $\mathcal{C}^A$ to denote the class of sets accepted by some $\mathcal{C}$ oracle machine with oracle A. For classes $\mathcal{C}$ and $\mathcal{D}$ of sets, let $\mathcal{C}^{\mathcal{D}}$ be $\bigcup_{D\in\mathcal{D}} \mathcal{C}^D$.

For each $k \in \mathbb{N}$, the notation "$[k]$" denotes a restriction of at most k oracle queries (in a sequential—i.e., "adaptive" or "Turing"—fashion). Restrictions of the number of oracle queries allowed may also depend on the input length. For example, $\mathrm{P}^{\mathrm{NP}[\log]}$ is the class of problems solvable by a DPOTM that, on inputs of length n, is allowed to access its NP oracle at most $\log n$ times. The notation "$[\mathcal{O}(1)]$" (as in $\mathrm{P}^{\mathrm{NP}[\mathcal{O}(1)]}$) denotes that, for some constant k, a "$[k]$" restriction holds.

[2] The concept being due to Toda [Tod91a] and the notation being due to Vollmer [Vol94], see the discussion in [HV95].

Definition 2.2.3 *1.* [KL80] *For any language class $\mathcal{C}$, let* $\mathcal{C}$/poly *be the class of all languages L for which there exist a set $A \in \mathcal{C}$, a polynomial p, and an advice function $h : \Sigma^* \to \Sigma^*$ such that for each length n, $|h(1^n)| = p(n)$, and for every $x \in \Sigma^*$, $x \in L$ if and only if $\langle x, h(1^{|x|})\rangle \in A$.*

2. For any function class $\mathcal{F}$, let $\mathcal{F}$/poly *be the class of all functions g for which there exist a function $f \in \mathcal{F}$, a polynomial p, and an advice function $h : \Sigma^* \to \Sigma^*$ such that for each length n, $|h(1^n)| = p(n)$, and for every $x \in \Sigma^*$, $g(x) = f(\langle x, h(1^{|x|})\rangle)$.*

Definition 2.2.4 *1. Given any two sets $A, B \subseteq \Sigma^*$, we say A* polynomial-time many-one reduces *to B (in symbols, $A \leq^{\mathrm{p}}_{\mathrm{m}} B$) if and only if there exists a function $f \in$ FP such that for all $x \in \Sigma^*$, $x \in A \iff f(x) \in B$.*

2. We say A polynomial-time Turing reduces *to B (in symbols, $A \leq^{\mathrm{p}}_{\mathrm{T}} B$) if and only if $A \in \mathrm{P}^B$.*

For other reducibilities, such as polynomial-time truth-table reducibility, we refer to the standard source [LLS75].

For any complexity class $\mathcal{C}$ and any set $B \subseteq \Sigma^*$, we say B is $\leq^{\mathrm{p}}_{\mathrm{m}}$*-hard* (respectively, $\leq^{\mathrm{p}}_{\mathrm{T}}$*-hard*) for $\mathcal{C}$ if for all sets $A \in \mathcal{C}$, $A \leq^{\mathrm{p}}_{\mathrm{m}} B$ (respectively, $A \leq^{\mathrm{p}}_{\mathrm{T}} B$). We say B is $\leq^{\mathrm{p}}_{\mathrm{m}}$*-complete* for $\mathcal{C}$ if B is $\leq^{\mathrm{p}}_{\mathrm{m}}$*-hard* for $\mathcal{C}$ and $B \in \mathcal{C}$. In all chapters except Chapter 6, we use the terms "$\mathcal{C}$-hard" and "$\mathcal{C}$-complete" to mean "$\leq^{\mathrm{p}}_{\mathrm{m}}$-hard for $\mathcal{C}$" and "$\leq^{\mathrm{p}}_{\mathrm{m}}$-complete for $\mathcal{C}$," respectively.[3]

We now provide the definitions of various other important notions of complexity theory.

Definition 2.2.5 *1. For any complexity class $\mathcal{C}$, a set L is said to be* $\mathcal{C}$-immune *if L is infinite and no infinite subset of L is in $\mathcal{C}$. Let $\mathcal{C}$-immune denote the class of all $\mathcal{C}$-immune sets.*

2. A set L is said to be $\mathcal{C}$-bi-immune *if both L and $\overline{L}$ are $\mathcal{C}$-immune. Let $\mathcal{C}$-bi-immune denote the class of all $\mathcal{C}$-bi-immune sets.*

3. For classes $\mathcal{C}$ and $\mathcal{D}$ of sets, $\mathcal{D}$ is said to be $\mathcal{C}$-immune (respectively, $\mathcal{C}$-bi-immune) if $\mathcal{D} \cap (\mathcal{C}\text{-immune}) \neq \emptyset$ (respectively, if $\mathcal{D} \cap (\mathcal{C}\text{-bi-immune}) \neq \emptyset$).

Definition 2.2.6 [BH77] *A bijection $\phi : \Sigma^* \to \Sigma^*$ is a* P-isomorphism *if ϕ is computable and invertible in polynomial time.*

[3]Note, however, the convention regarding "$\mathcal{C}$-hardness of counting properties of circuits" that will be made in Definition 6.1.3 on page 86.

Definition 2.2.7 [GS91] *The* ranking function *of a language $A \subseteq \Sigma^*$ is the function $r : \Sigma^* \to \mathbb{N}$ that maps each $x \in \Sigma^*$ to $||\{y \leq x \mid y \in A\}||$. A language A is* rankable *if its ranking function is computable in polynomial time.*

That is, the ranking function for A tells us the number of strings in A up to a given string. To avoid confusion, we mention that the notion of rankability used here is sometimes called "P-rankability" (e.g., in [RH96, HRW97b]), and is also sometimes referred to as "strong P-rankability" (e.g., in [HR90]).

For any set L, the *census function* of L, $census_L : \Sigma^* \to \mathbb{N}$, is defined by $census_L(1^n) = ||L^{=n}||$; see Section 8.2 for discussion of this definition. A set S is said to be *sparse* if there is a polynomial p such that for each length n, $census_S(1^n) \leq p(n)$. A set T is said to be *tally* if $T \subseteq \{1\}^*$.

Definition 2.2.8 [HY84] *A set S is* P-printable *if there exists a DPTM M such that for each length n, M on input 1^n prints all elements of S having length at most n.*

P-printability [HY84] is a notion arising in a variety of contexts, including the theory of Kolmogorov complexity, data compression, and P-uniform circuit complexity. Hence, there are numerous characterizations of P-printability known. For instance, Allender and Rubinstein [AR88] proved that the following four statements are equivalent: (1) S is P-printable; (2) S is sparse and rankable; (3) S is P-isomorphic to some tally set in P; (4) S is a set in P having small generalized (unconditional) Kolmogorov complexity.[4]

We now present the notion of one-way function we will be dealing with in Chapter 4; see also Definition 4.2.1 on page 46, which gives a variant of Definition 2.2.9 below that is tailored to the case of *binary* functions. Note that we discuss one-way functions in the complexity-theoretic setting introduced by Grollmann and Selman [GS88], see also, e.g., [Ko85, Sel92, RS97]. So-called cryptographic one-way functions are not discussed here, though we should mention that one-way functions, and particularly one-way permutations, have been interestingly studied in that context, see, e.g., the papers [Yao82, IR89, HILL91].

Definition 2.2.9

1. *A function f is* honest *if there is a polynomial p such that for every $y \in$ image(f) there exists a string $x \in$ domain(f) such that $y = f(x)$ and $|x| \leq p(|y|)$.*

2. *A function f is* poly-to-one *if there exists a polynomial p such that $||f^{-1}(y)|| \leq p(|y|)$ for each $y \in$ image(f).*

[4] Regarding item (4), the precise meaning of those terms will be made clear in Section 3.2, where the notion of Kolmogorov complexity is presented, see Definition 3.2.5.

3. *A (many-to-one) function f is said to be* FP-invertible *if there exists a function $g \in$ FP such that for every $y \in$ image(f), $g(y)$ prints* some *value of $f^{-1}(y)$. In particular, if f is one-to-one,* FP-*invertibility of f means $f^{-1} \in$ FP.*

4. *A function f is said to be a* one-way function *if f is honest, $f \in$ FP, and f is not* FP-*invertible.*[5]

5. *If $f : \Sigma^* \to \Sigma^*$ is a surjective and one-to-one one-way function, f is called a* one-way permutation.

Note that the honesty of one-way functions is required in order to avoid the case that the FP-noninvertibility is trivial; see also Footnote 5 on page 59 for a different notion of honesty.

2.3 Boolean Functions and Circuits

An n-ary *Boolean function* is a mapping f_n from $\{0,1\}^n$ to $\{0,1\}$. Some specific Boolean functions will be defined in Chapter 7.

Circuits built over Boolean gates are ways of representing Boolean functions.[6] An AND (respectively, OR) gate outputs 1 (respectively, 0) if and only if all its inputs are 1 (respectively, 0), where we allow AND and OR gates to have unbounded fanin unless stated otherwise. A negation gate outputs $1 - i$ if its input has value $i \in \{0,1\}$. In Chapter 7, we will also consider circuits having (unbounded fanin) PARITY gates. A PARITY gate, denoted $\oplus$, outputs 1 if and only if an odd number of its inputs are 1. Circuits are assumed to be encoded in some standard way—in fact, for simplicity of expression, we will often treat a circuit and its encoding as interchangeable.

The *size* of a circuit is the number of its gates. The *circuit complexity* (or *size*) of a Boolean function f is the size of a smallest circuit computing f.[7] The *depth* of a circuit is the length of a longest path from its input gates to its output gate.

Given an arity n circuit C, $\#(C)$ denotes under how many of the 2^n possible input patterns C evaluates to 1.

[5] According to our general convention regarding functions, all one-way functions may potentially be many-to-one and partial, unless explicitly stated as being one-to-one, poly-to-one, or total.

[6] More precisely, we are dealing with *families* of Boolean functions, one function for each arity n, that are realized by *circuit families*. By convention, when we speak of "a" circuit C computing "a" function f, we implicitly mean a family $C = (C_n)_{n\in\mathbb{N}}$ of circuits computing a family $f = (f_n)_{n\in\mathbb{N}}$ of functions (i.e., for each n, C_n is a circuit with n input gates and one output gate that outputs the value $f_n(x)$ for each $x \in \{0,1\}^n$).

[7] For families of Boolean circuits or functions, the size is a function of n, where n is the number of inputs.

Chapter 3

Easy Sets and Hard Certificate Schemes

3.1 Introduction

Borodin and Demers [BD76] proved the following result.

Theorem 3.1.1 [BD76] *If* P $\neq$ NP $\cap$ coNP*, then there exists a set* L *such that*

1. $L \in$ P,
2. $L \subseteq$ SAT, *and*
3. *For no polynomial-time computable function* f *does it hold that: for each* $F \in L$, $f(F)$ *outputs a satisfying assignment of* F.

That is, under a hypothesis most complexity theoreticians would guess to be true, it follows that there is a set of satisfiable formulas for which it is trivial to determine they are satisfiable, yet it is hard to determine why (i.e., via what satisfying assignment) they are satisfiable.

Motivated by their work, the present chapter seeks to study, complexity-theoretically, the classes of sets that do or do not have easy certificates. In particular, we are interested in the following four classes. Any nondeterministic polynomial-time Turing machine M is said to *always have easy certificates* (respectively, to *infinitely often have easy certificates*) if there is some function f computable in polynomial time such that for all (respectively, for infinitely many) inputs $x \in L(M)$, the value $f(x)$ is an accepting path of M on input x. $\text{EASY}_{\forall}^{\forall}$ (respectively, $\text{EASY}_{\forall}^{\exists}$) is the class of all NP sets L such that each NP machine (respectively, some NP machine) accepting L always has easy certificates. $\text{EASY}_{\text{io}}^{\forall}$ (respectively, $\text{EASY}_{\text{io}}^{\exists}$) is the class of all NP sets L such that each NP machine (respectively,

some NP machine) accepting L infinitely often has easy certificates. Thus, the above result of Borodin and Demers can in this notation be rephrased as: If P $\neq$ NP $\cap$ coNP then P $\not\subseteq$ EASY$^{\forall}_{\forall}$.

However, we note that EASY$^{\exists}_{\forall}$ = P and EASY$^{\exists}_{\text{io}}$ equals the class of non-P-immune NP sets. The main focus of the present chapter is thus on the classes EASY$^{\forall}_{\forall}$ and EASY$^{\forall}_{\text{io}}$.

In Section 3.2, we formally define our notations and establish the inclusion relations between the four above-mentioned classes. We also provide equivalent characterizations of the classes EASY$^{\forall}_{\forall}$ and EASY$^{\forall}_{\text{io}}$ in terms of relative generalized Kolmogorov complexity, thus showing that they are robust.

In Section 3.3, we provide structural conditions—regarding immunity, P-printability, and class collapses—that put upper and lower bounds on the sizes of EASY$^{\forall}_{\forall}$ and EASY$^{\forall}_{\text{io}}$. Among the results we establish are the implications: (a) if NP $\cap$ coNP has P-bi-immune sets, then EASY$^{\forall}_{\text{io}}$ = FINITE; (b) if NP has P-immune sets, then EASY$^{\forall}_{\text{io}} \neq$ NP; and (c) if EASY$^{\forall}_{\text{io}} \neq$ NP, then there exists an infinite P set having no infinite P-printable subset. We also prove equivalences between such conditions, e.g., P $\neq$ NP if and only if EASY$^{\forall}_{\forall} \neq$ NP.

Finally, in Section 3.4, we provide negative results showing that some of our positive claims are optimal with regard to techniques that relativize. Our negative results are proven using a novel observation: We show that the classical "wide spacing" oracle construction technique yields instant non-bi-immunity results. Furthermore, we establish a result that improves upon Baker, Gill, and Solovay's classical result that NP $\neq$ P = NP $\cap$ coNP holds in some relativized world [BGS75], and that in addition links their result with the above-stated result of Borodin and Demers.

3.2 Definitions and Robustness

As a notational convention, for any NPTM N, we will say that N *always has easy certificates* (respectively, that N *infinitely often has easy certificates*) if (the encoding of) an accepting path of $N(x)$ can be printed in polynomial time for each string $x \in L(N)$ (respectively, for infinitely many strings $x \in L(N)$). Similarly, N is said to *only have hard certificates* (respectively, to *infinitely often have hard certificates*) if no FP function is able to output (the encoding of) an accepting path of $N(x)$ for each string $x \in L(N)$ (respectively, for infinitely many strings $x \in L(N)$).

Definition 3.2.1 *Let $L \subseteq \Sigma^*$ be any set in* NP.

1. $L \in$ EASY$^{\forall}_{\forall}$ *if and only if*

$$(\forall \textit{NPTM } N \,:\, L(N) = L)\,(\exists f_N \in \text{FP})\,(\forall x \in L)\,[f_N(x) \in \text{ACC}_N(x)].$$

2. $L \in \mathrm{EASY}^{\forall}_{\mathrm{io}}$ *if and only if either* L *is finite, or*

$$(\forall \textit{NPTM } N \ : \ L(N) = L)\,(\exists f_N \in \mathrm{FP})\,(\exists^{io} x \in L)\,[f_N(x) \in \mathrm{ACC}_N(x)].$$

3. $L \in \mathrm{EASY}^{\exists}_{\forall}$ *if and only if*

$$(\exists \textit{NPTM } N)\,[L(N) = L \ \wedge \ (\exists f_N \in \mathrm{FP})\,(\forall x \in L)\,[f_N(x) \in \mathrm{ACC}_N(x)]].$$

4. $L \in \mathrm{EASY}^{\exists}_{\mathrm{io}}$ *if and only if either* L *is finite, or*

$$(\exists \textit{NPTM } N)\,[L(N) = L \ \wedge \ (\exists f_N \in \mathrm{FP})\,(\exists^{io} x \in L)\,[f_N(x) \in \mathrm{ACC}_N(x)]].$$

Remark 3.2.2 *1. It is easy to see that both* $\mathrm{EASY}^{\forall}_{\forall}$ *and* $\mathrm{EASY}^{\exists}_{\forall}$ *contain all finite sets. On the other hand, the* $\mathrm{EASY}^{\cdots}_{\mathrm{io}}$ *classes are* defined *so as to also contain all finite sets; this is just for uniformity and since we feel that it is reasonable to require that the finite sets satisfy any suggested notion of "easy sets."*

2. Note that we can analogously define $\mathrm{EASY}^{\forall}_{\mathrm{ae}}$ *and* $\mathrm{EASY}^{\exists}_{\mathrm{ae}}$ *by using the quantification "*$\forall^{ae} x \in L$*" rather than "*$\forall x \in L$*" in part 1 and part 3 of the above definition. However, since the classes* $\mathrm{EASY}^{\forall}_{\forall}$ *and* $\mathrm{EASY}^{\exists}_{\forall}$ *(as are most complexity classes) are closed under finite variations, it is clear that* $\mathrm{EASY}^{\forall}_{\mathrm{ae}} = \mathrm{EASY}^{\forall}_{\forall}$ *and* $\mathrm{EASY}^{\exists}_{\mathrm{ae}} = \mathrm{EASY}^{\exists}_{\forall}$*. Moreover, we show below that* $\mathrm{EASY}^{\exists}_{\forall} = \mathrm{P}$ *and that* $\mathrm{EASY}^{\exists}_{\mathrm{io}}$ *equals the class of all non-*P*-immune* NP *sets, and we therefore will not further discuss these two classes in this thesis.*

Theorem 3.2.3 states the inclusion relations between FINITE, NP, and the above-defined four classes of easy NP sets, see Figure 3.1. These inclusions follow immediately from Definition 3.2.1.

Theorem 3.2.3 *The following two statements are true.*

1. $\mathrm{FINITE} \subseteq \mathrm{EASY}^{\forall}_{\forall} \subseteq \mathrm{EASY}^{\forall}_{\mathrm{io}} \subseteq \mathrm{EASY}^{\exists}_{\mathrm{io}} \subseteq \mathrm{NP}$.

2. $\mathrm{EASY}^{\forall}_{\forall} \subseteq \mathrm{EASY}^{\exists}_{\forall} \subseteq \mathrm{EASY}^{\exists}_{\mathrm{io}}$.

Now we characterize the classes $\mathrm{EASY}^{\exists}_{\forall}$ and $\mathrm{EASY}^{\exists}_{\mathrm{io}}$.

Theorem 3.2.4 *The following two statements are true.*

1. $\mathrm{EASY}^{\exists}_{\forall} = \mathrm{P}$.

2. $\mathrm{EASY}^{\exists}_{\mathrm{io}} = \overline{\text{P-immune}} \cap \mathrm{NP}$.

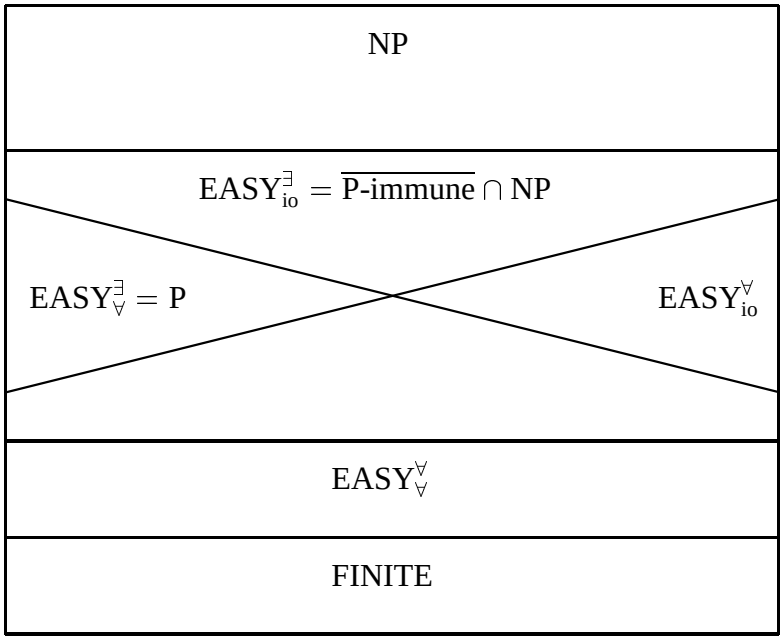

Figure 3.1: Inclusions between classes of NP sets having easy certificates

Proof. (1) The inclusion $\text{P} \subseteq \text{EASY}^{\exists}_{\forall}$ holds by definition. The converse inclusion, $\text{EASY}^{\exists}_{\forall} \subseteq \text{P}$, follows from the fact that, for each NPTM N, $\bigcup_{x \in \Sigma^*} \text{ACC}_N(x)$ is a set in P. More precisely, let L be any set in $\text{EASY}^{\exists}_{\forall}$, and let this membership be witnessed by NPTM N and FP function f_N. There exists a DPTM M that recognizes L as follows. On input x, M computes $f_N(x)$, checks whether or not $f_N(x) \in \text{ACC}_N(x)$ by simulating the corresponding computation path of $N(x)$, and accepts or rejects accordingly. If $x \in L$, then $f_N(x) \in \text{ACC}_N(x)$, and M accepts x. If $x \notin L$, then $f_N(x)$ cannot be an accepting path of $N(x)$, and thus M rejects x.

(2) Let $L \in \text{EASY}^{\exists}_{\text{io}}$ via NPTM N and FP function f_N. Thus, $L \in$ NP, and if L is not a finite set, then the set $\{x \in L \mid f_N(x) = y \text{ and } N(x) \text{ accepts on path } y\}$ is an infinite subset of L that is in P. Hence, L is not P-immune.

Conversely, let A be any NP set that is not P-immune. Let M_A be an NPTM accepting A. If A is finite, then we are done. So suppose A is infinite, and there is an infinite set B such that $B \subseteq A$ and $B \in$ P via DPTM M_B. We now describe an NPTM N and an FP function f_N that witness $A \in \text{EASY}^{\exists}_{\text{io}}$. On input x, N first simulates the computation of $M_B(x)$, and accepts x if M_B accepts x. If M_B rejects x, then N simulates the computation of $M_A(x)$. It holds that $L(N) = A$. For each x, define $f_N(x)$ to be the (suitably encoded) computation of $M_B(x)$. Since B is an infinite set, $f_N(x)$ prints an accepting path

of $N(x)$ for infinitely many $x \in A$. ∎

The Kolmogorov complexity of finite strings was introduced independently by Kolmogorov [Kol65] and Chaitin [Cha66]. Roughly speaking, the Kolmogorov complexity of a finite binary string x is the length of a shortest program that generates x. Intuitively, if a string x can be generated by a program shorter than x itself, then x can be "compressed." The notion of *generalized* Kolmogorov complexity ([Adl79, Har83a, Sip83b], see Li and Vitányi [LV93, LV90] for a nice survey of the field) is a version of Kolmogorov complexity that provides information about not only whether and how far a string can be *compressed*, but also how fast it can be *restored*. We now give the definition of (unconditional and conditional) generalized Kolmogorov complexity.

Definition 3.2.5 ([Har83a], see also [Adl79, Sip83b]) *For any Turing machine T and functions s and t mapping $\mathbb{N}$ to $\mathbb{N}$, define $\mathrm{K}_T[s(n), t(n)]$ to be the set*

$$\{x \mid (\exists y)\,[\,|x| = n \text{ and } |y| \leq s(n) \text{ and } T(y) \text{ outputs } x \text{ in at most } t(n) \text{ steps}\,]\}.$$

It was shown in [Har83a] that there exists a *universal*[1] Turing machine U such that for any other Turing machine T there exists a constant c such that

$$\mathrm{K}_T[s(n), t(n)] \subseteq \mathrm{K}_U[s(n) + c,\ c\,t(n)\log t(n) + c].$$

Fixing a universal Turing machine U and dropping the subscript, the *unconditional* generalized Kolmogorov complexity will be denoted by $\mathrm{K}[s(n), t(n)] = \mathrm{K}_U[s(n), t(n)]$. The *conditional* generalized Kolmogorov complexity (under condition z), in which the information of the string z is given for free and does not count for the complexity, is defined as follows.

Definition 3.2.6 *Let U be a fixed universal Turing machine and z be a string. For functions s and t mapping $\mathbb{N}$ to $\mathbb{N}$, define $\mathrm{K}[s(n), t(n) \mid z]$ to be the set*

$$\{x \mid (\exists y)\,[\,|x| = n \text{ and } |y| \leq s(n) \text{ and } U(\langle y, z\rangle) \text{ outputs } x \text{ in} \leq t(n) \text{ steps}\,]\}.$$

In particular, $\mathrm{K}[s(n), t(n) \mid \epsilon] = \mathrm{K}[s(n), t(n)]$.

Of particular interest in this chapter are certificates (i.e., strings encoding accepting computation paths of NPTMs) that have *small* generalized Kolmogorov complexity.

[1] Roughly speaking, a universal Turing machine U expects as input a pair of a (suitably encoded) Turing machine T and an input string y and simulates the computation of $T(y)$. More precisely, denoting the encoding of T by $\mathrm{code}(T)$ and using our pairing function, U runs on input $\langle \mathrm{code}(T), y\rangle$ and outputs the result of the computation of $T(y)$.

Levin (see [Tra84]) and Adleman [Adl79] independently discovered the connection between small generalized Kolmogorov complexity and certificates. This connection has also been used in other contexts ([HW91], see also [HR90, GT91] and the comments in [HR90] on [CH89]).

The notion of P-printability is presented in Definition 2.2.8 on page 17. The P-printable sets are closely related to sets of strings having small unconditional generalized Kolmogorov complexity: A set S is P-printable if and only if $S \in \mathrm{P}$ and $S \subseteq \mathrm{K}[k \log n, n^k]$ for some constant k ([AR88], see also [BB86, HH88b, Rub86]). Below we note a similar connection between the sets in $\mathrm{EASY}_{\forall}^{\forall}$ and $\mathrm{EASY}_{\mathrm{io}}^{\forall}$ and the sets of certificates having small *conditional* generalized Kolmogorov complexity, thus showing the robustness of these notions. Due to Theorem 3.2.4, the corresponding claims for $\mathrm{EASY}_{\forall}^{\exists}$ and $\mathrm{EASY}_{\mathrm{io}}^{\exists}$ are omitted. Though the flavor of the correspondence here invoked is by now standard (see, e.g., the above papers), we include the proof of Observation 3.2.7 for completeness.

Observation 3.2.7 *The following two statements are equivalent.*

1. $L \in \mathrm{EASY}_{\forall}^{\forall}$.
2. *For each normalized NPTM N accepting L there is a constant k (which may depend on N) such that for each string $x \in L$ it holds that*

$$\mathrm{ACC}_N(x) \cap \mathrm{K}[k \log n, n^k \mid x] \neq \emptyset.$$

Proof. In one direction the function proving a machine easy itself yields Kolmogorov-simple certificates. That is, for any normalized NPTM N accepting the given $\mathrm{EASY}_{\forall}^{\forall}$ set L, there is an FP function f_N that outputs an accepting path of $N(x)$ for each $x \in L$. Note that for each $x \in L$, the program for f_N, encoded as a string y, has constant size, and the universal Turing machine U, running on input $\langle y, x \rangle$, can generate $f_N(x)$ in time polynomial in $|f_N(x)|$. Hence, for each $x \in L$, the certificate $f_N(x)$ is in $\mathrm{K}[k \log n, n^k \mid x]$ for some constant k depending only on f_N, and thus on N.

In the other direction, let N be any NPTM accepting L. By assumption, for each $x \in L$, $N(x)$ has certificates of small conditional Kolmogorov complexity relative to x, i.e., $N(x)$ has certificates in $\mathrm{K}[k \log n, n^k \mid x]$ for some constant k. Note that n, the length of those certificates, is polynomial in $|x|$; let p be some such polynomial bound. So, for each x, $n = p(|x|)$ is a polynomial bound on the length of the certificates of $N(x)$. There are at most $2^{\mathcal{O}(\log n)} = n^{\ell}$ (for some suitable constant ℓ) short strings that potentially encode programs y such that the universal Turing machine U, running on input $\langle y, x \rangle$, produces a certificate of $N(x)$ in time polynomial in n, say in time $n^m = (p(|x|))^m$. Let $q(|x|) = \max\{(p(|x|))^{\ell}, (p(|x|))^m\}$.

The function f_N proving N easy works on input x as follows. In a brute-force manner, f_N runs the universal machine on the pairs $\langle y, x\rangle$ for all the at most $q(|x|)$ many short strings y, $|y| \leq k \log n$, for $q(|x|)$ steps, and then for each output checks if the output is an accepting path of $N(x)$, and eventually outputs the first such accepting path found. If no accepting path was found, the input x is not in L. Note that f_N is polynomial-time computable and, for each input $x \in L$, outputs a certificate of $N(x)$. ∎

Remark 3.2.8 *Note that the normalization requirement in the above observation is crucial, since our definition of conditional generalized Kolmogorov complexity displays the strange feature that for machines that are not normalized, if we use a certain simple polynomial-time computable and polynomial-time invertible pairing function, say $\langle \cdot, \cdot \rangle_{weird}$, to encode the pair of the program y and the condition z as input $\langle y, z\rangle_{weird}$ to the universal Turing machine, then even the empty string has* non-constant *conditional Kolmogorov complexity. Due to our normalization requirement, however, this issue is not germane here.*

The proof of Observation 3.2.9 follows precisely the lines of the proof of Observation 3.2.7.

Observation 3.2.9 *The following two statements are equivalent.*

1. $L \in \mathrm{EASY}^{\forall}_{\mathrm{io}}$.
2. *For each normalized NPTM N accepting L there is a constant k (which may depend on N) such that for infinitely many strings $x \in L$ it holds that*

$$\mathrm{ACC}_N(x) \cap \mathrm{K}[k \log n, n^k \mid x] \neq \emptyset.$$

3.3 Positive Results

In this section, we prove a number of implications between certain properties of subclasses of NP that are summarized in Figure 3.2. Usually, when one is trying to give strong evidence for some complexity-theoretic statement S not to be true, one does so by showing that S implies P = NP. In contrast, Figure 3.2 has P ≠ NP as its top conclusion. Nonetheless, the implications of Figure 3.2 are not meaningless. Their importance is obvious in light of the fact that the statements of the figure are well-studied and important conditions in complexity theory. The implications of Figure 3.2 state that these conditions are at least as hard to prove as proving P ≠ NP, and they explore the logical fine-structure among those important conditions.

Note that no chain of implications in Figure 3.2 that contains an arrow marked by a "$*$" is invertible in all relativized worlds.

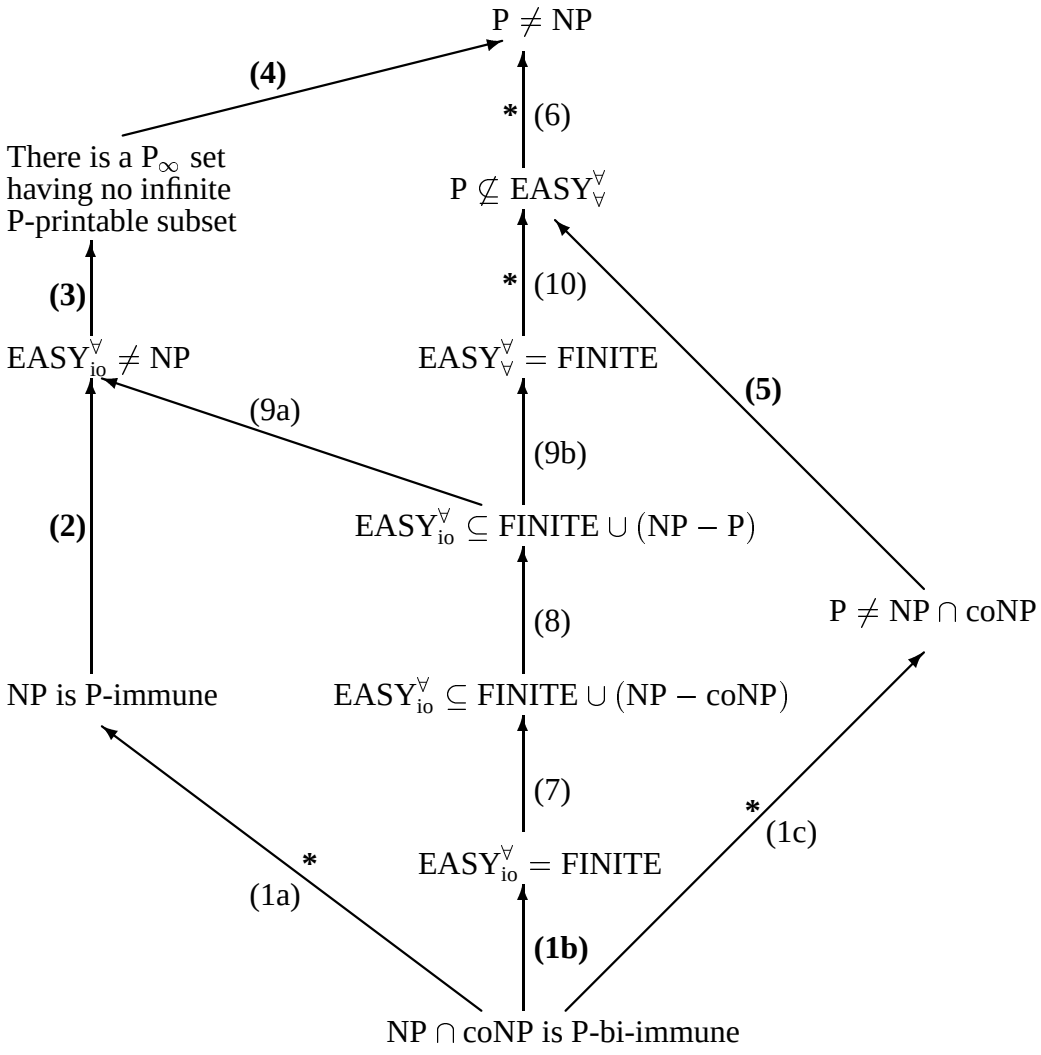

Figure 3.2: Implications between various properties of (classes of) sets within NP

Key: Implications represented by arrows that are marked by a "$*$" are not invertible up to the limits of relativizations, see Section 3.4. Arrows labeled by boldface numbers indicate nontrivial implications to be proven in Theorem 3.3.3.

We first discuss the trivial implications of Figure 3.2. We stress that these trivial statements are included not only in order to make the picture displayed in Figure 3.2 as complete as possible, but also for the following reason. In the next section, we will prove that the reverse of some implication chains comprising both trivial and more interesting implications not only fails in suitable relativized worlds, but, even worse, this relativized failure can already be shown for the trivial part of the implication chain considered. Therefore, it does make sense to explicitly state such trivial implications.

These trivial facts are either immediately clear, or they follow from simple set-theoretical arguments, or they are straightforwardly established by the equivalences given in Theorem 3.3.1 below. For instance, the equivalence of "$\mathrm{EASY}^{\forall}_{\forall} = \mathrm{FINITE}$" and "$\mathrm{EASY}^{\forall}_{\forall} \subseteq \mathrm{FINITE} \cup (\mathrm{NP} - \mathrm{P})$" can be seen by simple set-theoretical considerations.[2] The statement "$\mathrm{EASY}^{\forall}_{\forall} = \mathrm{FINITE}$," in turn, immediately implies the statement "$\mathrm{P} \not\subseteq \mathrm{EASY}^{\forall}_{\forall}$," see arrow (10) in Figure 3.2. We have been informed that the authors of [FFNR96] have shown a number of very interesting conditions, including "$\Sigma^* \notin \mathrm{EASY}^{\forall}_{\forall}$" and "there exists an honest polynomial-time computable *onto* function that is not polynomial-time invertible," to be all equivalent to the statement "$\mathrm{P} \not\subseteq \mathrm{EASY}^{\forall}_{\forall}$."

Furthermore, the arrows in Figure 3.2 labeled (1a), (1c), (7), and (8) are immediately clear. Concerning arrows (9a) and (9b), note that (9b) follows from the equivalence of "$\mathrm{EASY}^{\forall}_{\forall} = \mathrm{FINITE}$" and "$\mathrm{EASY}^{\forall}_{\forall} \subseteq \mathrm{FINITE} \cup (\mathrm{NP} - \mathrm{P})$" stated in the previous paragraph, whereas (9a) is implied by part 2 of Theorem 3.3.1 below. Similarly, arrow (6) holds due to part 1 of Theorem 3.3.1, since if $\mathrm{P} \not\subseteq \mathrm{EASY}^{\forall}_{\forall}$, then there exists a set in $\mathrm{NP} - \mathrm{EASY}^{\forall}_{\forall}$, and thus we have $\mathrm{P} \neq \mathrm{NP}$.

The following proposition gives characterizations for two nodes of Figure 3.2.

Theorem 3.3.1 *The following two statements are true.*

1. $\mathrm{P} \neq \mathrm{NP}$ *if and only if* $\mathrm{EASY}^{\forall}_{\forall} \neq \mathrm{NP}$.

2. $\mathrm{EASY}^{\forall}_{\mathrm{io}} \subseteq \mathrm{FINITE} \cup (\mathrm{NP} - \mathrm{P})$ *if and only if* $\Sigma^* \notin \mathrm{EASY}^{\forall}_{\mathrm{io}}$.

Proof. (1) Adleman ([Adl79], see also [Tra84] for a discussion of Levin's related work) has shown that $\mathrm{P} = \mathrm{NP}$ if and only if for each normalized NPTM M there is a constant k such that for each string $x \in L(M)$ it holds that $\mathrm{ACC}_M(x) \cap \mathrm{K}[k \log n, n^k \mid x] \neq \emptyset$. By Observation 3.2.7, this equivalence implies that $\mathrm{P} = \mathrm{NP}$ if and only if $\mathrm{EASY}^{\forall}_{\forall} = \mathrm{NP}$.

(2) First note that the statement "$\mathrm{EASY}^{\forall}_{\mathrm{io}} \subseteq \mathrm{FINITE} \cup (\mathrm{NP} - \mathrm{P})$" is equivalent to the statement "$\mathrm{EASY}^{\forall}_{\mathrm{io}} \cap \mathrm{P}_{\infty} = \emptyset$," and thus immediately implies $\Sigma^* \notin \mathrm{EASY}^{\forall}_{\mathrm{io}}$, since $\Sigma^* \in \mathrm{P}_{\infty}$.

[2] To be definite, for all sets A, B, C, and X, if $A \subseteq X \subseteq B \subseteq C$, then $(X = A \iff X \subseteq A \cup (C - B))$. Taking $A = \mathrm{FINITE}$, $B = \mathrm{P}$, $C = \mathrm{NP}$, and $X = \mathrm{EASY}^{\forall}_{\forall}$, we have verified our claim.

For the converse implication, assume there exists a set L in $\mathrm{EASY}^{\forall}_{\mathrm{io}} \cap \mathrm{P}_{\infty}$. Let M_L be a DPTM such that $L(M_L) = L$. We show that $\Sigma^* \in \mathrm{EASY}^{\forall}_{\mathrm{io}}$. Let N be any NPTM such that $L(N) = \Sigma^*$. By way of contradiction, suppose N has easy certificates only for finitely many $x \in \Sigma^*$. Consider the following NPTM N_L for L. On input x, N_L first simulates the computation of $N(x)$, and then, for every path of this simulation, N_L simulates $M_L(x)$ and accepts accordingly. Note that $L(N_L) = L$. By our supposition that N has easy certificates for finitely many inputs only, N_L also can have easy certificates for at most finitely many inputs, contradicting that $L \in \mathrm{EASY}^{\forall}_{\mathrm{io}} \cap \mathrm{P}_{\infty}$. Hence $\Sigma^* \in \mathrm{EASY}^{\forall}_{\mathrm{io}}$, completing the proof. ∎

The first part of Theorem 3.3.1 along with the first part of Theorem 3.2.4 yield the following interesting consequence. Corollary 3.3.2 can be interpreted as saying that for any NP-complete set to have *only* easy certificate schemes, it suffices that it has *some* easy certificate scheme. Putting it differently, Corollary 3.3.2 is a downward collapse result of sorts, since it says that the collapse of NP to a potentially smaller class, $\mathrm{P} = \mathrm{EASY}^{\exists}_{\forall}$, implies the collapse of NP to $\mathrm{EASY}^{\forall}_{\forall}$, which itself is potentially even smaller a class than P.

Corollary 3.3.2 $\mathrm{NP} = \mathrm{EASY}^{\exists}_{\forall}$ *if and only if* $\mathrm{NP} = \mathrm{EASY}^{\forall}_{\forall}$.

Next, we prove the nontrivial implications of Figure 3.2.

Theorem 3.3.3 *The following five statements are true.*

1. *If* $\mathrm{NP} \cap \mathrm{coNP}$ *is* P*-bi-immune, then* $\mathrm{EASY}^{\forall}_{\mathrm{io}} = \mathrm{FINITE}$.
2. *If* NP *is* P*-immune, then* $\mathrm{EASY}^{\forall}_{\mathrm{io}} \neq \mathrm{NP}$.
3. *If* $\mathrm{EASY}^{\forall}_{\mathrm{io}} \neq \mathrm{NP}$, *then there exists an infinite* P *set having no infinite* P*-printable subset.*
4. [All92] *If there exists an infinite* P *set having no infinite* P*-printable subset, then* $\mathrm{P} \neq \mathrm{NP}$.
5. [BD76] *If* $\mathrm{NP} \cap \mathrm{coNP} \neq \mathrm{P}$, *then* $\mathrm{P} \not\subseteq \mathrm{EASY}^{\forall}_{\forall}$.

Proof. (1) Let Q be any P-bi-immune set such that $Q \in \mathrm{NP} \cap \mathrm{coNP}$ via NPTMs N_Q and $N_{\overline{Q}}$, that is, $L(N_Q) = Q$ and $L(N_{\overline{Q}}) = \overline{Q}$. By way of contradiction, assume there exists an infinite set L in $\mathrm{EASY}^{\forall}_{\mathrm{io}}$. Let N be any NPTM accepting L. Consider the following NPTM $\widehat{N}$ for L. Given x, $\widehat{N}$ runs $N(x)$ and rejects on all rejecting paths of $N(x)$. On all accepting paths of $N(x)$, $\widehat{N}$ nondeterministically guesses whether $x \in Q$ or $x \in \overline{Q}$, simultaneously guessing certificates (i.e., accepting paths of $N_Q(x)$ or $N_{\overline{Q}}(x)$) for whichever guess was

made, and accepts on each accepting path of $N_Q(x)$ or $N_{\overline{Q}}(x)$. Note that $L(\widehat{N}) = L$. By our assumption that L is an infinite set in $\text{EASY}^{\forall}_{\text{io}}$, $\widehat{N}$ has easy certificates for infinitely many inputs. Let $f_{\widehat{N}}$ be an FP function that infinitely often outputs an easy certificate of $\widehat{N}$. Define

$$\widehat{L} = \{x \mid f_{\widehat{N}}(x) \text{ outputs an easy certificate of } \widehat{N}(x)\}.$$

Note that $\widehat{L}$ is an infinite subset of L, and that for any input x, it can be checked in polynomial time whether x belongs to $Q \cap \widehat{L}$ or $\overline{Q} \cap \widehat{L}$, respectively, by simply checking whether the string printed by $f_{\widehat{N}}$ indeed certifies either $x \in Q \cap \widehat{L}$ or $x \in \overline{Q} \cap \widehat{L}$. Thus, either $Q \cap \widehat{L}$ or $\overline{Q} \cap \widehat{L}$ must be an infinite set in P, which contradicts the hypothesis that Q is P-bi-immune. Hence, every set in $\text{EASY}^{\forall}_{\text{io}}$ is finite.

(2) Let L be any P-immune NP set. We claim that $L \notin \text{EASY}^{\forall}_{\text{io}}$. Suppose to the contrary that $L \in \text{EASY}^{\forall}_{\text{io}}$. Let N be any NPTM accepting L. Then there exists an FP function f_N such that $f_N(x) \in \text{ACC}_N(x)$ for infinitely many inputs x. Define

$$B = \{x \mid f_N(x) \in \text{ACC}_N(x)\}.$$

Note that B is an infinite subset of L and $B \in \text{P}$, contradicting the P-immunity of L.

(3) If $\text{EASY}^{\forall}_{\text{io}} \neq \text{NP}$, then there exist an infinite NP set L and an NPTM N accepting L such that

$$(\forall f \in \text{FP})\,(\forall^{\text{ae}} x \in L)\,[f(x) \notin \text{ACC}_N(x)]. \tag{3.3.1}$$

Let q be a polynomial such that $|\langle x, y\rangle| = q(|x|)$ for any string x and any path y of $N(x)$. Define

$$D = \{\langle x, y\rangle \mid y \in \text{ACC}_N(x)\}.$$

Note that D is an infinite set in P. Suppose there exists an infinite set A such that $A \subseteq D$ and A is P-printable via some DPTM M. Define an FP function f_A that is computed by DPTM M_A as follows. On input x, M_A simulates the computation of $M(1^{q(|x|)})$ and prints all elements of A up to length $q(|x|)$. If a string of the form $\langle x, y\rangle$ is printed, M_A outputs y. Note that $f_A(x) \in \text{ACC}_N(x)$ for infinitely many $x \in L$, contradicting (3.3.1) above. Hence, D has no infinite P-printable subset.

(4) This implication can be proven using results due to Allender [All92]. First, some definitions are needed. Let us consider another version of time-bounded Kolmogorov complexity, a version that is due to Levin [Lev84], see also [Lev73]. For the fixed universal Turing machine U and any string x, define $\text{Kt}(x)$ to be

$$\min\{|y| + \log n \mid U(y) \text{ outputs } x \text{ in at most } n \text{ steps}\}.$$

For any set L, let $\mathrm{K}_L(n) = \min\{\mathrm{Kt}(x) \mid x \in L^{=n}\}$.

An *NE predicate* is a relation R defined by an NE machine M: $R(x, y)$ is true if and only if y encodes an accepting path of $M(x)$. An NE predicate R is *E-solvable* if there is some function $f \in$ FE such that

$$(\forall x)\,[(\exists y)\,[R(x, y)] \iff R(x, f(x))].$$

Allender [All92] has shown the following two statements.

(a) There exists an infinite P set having no infinite P-printable subset if and only if there exists a set $B \in$ P such that $\mathrm{K}_B(n) \in \omega(\log n)$.

(b) There exists an NE predicate that is not E-solvable if and only if there exists a set $C \in$ P such that $\mathrm{K}_C(n) \notin \mathcal{O}(\log n)$.

Since $\mathrm{K}_B(n) \in \omega(\log n)$ implies $\mathrm{K}_B(n) \notin \mathcal{O}(\log n)$ and since the existence of an NE predicate that is not E-solvable implies P $\neq$ NP, (4) is proven.

We also give a more transparent direct proof of (4). To prove the contrapositive, assume P $=$ NP. Let L be any infinite set in P. Define the set

$$A = \{\langle 0^n, w\rangle \mid n \geq 0 \ \wedge\ |w| = n \ \wedge\ (\exists z \in L^{=n})\,[z < w]\}.$$

Note that $A \in$ NP. By our assumption, $A \in$ P. Define the set of the lexicographically smallest length n strings of L for each length n:

$$S = \{x \in L \mid (\forall y \in L^{=|x|})\,[x \leq y]\}.$$

Note that S is an infinite subset of L. Furthermore, S is P-printable, since we can find, at each length n, the lexicographically smallest length n string in L (which is the length n string of S) via prefix search that can be performed in $\mathrm{FP}^A = \mathrm{FP}$. Thus, every infinite set in P has an infinite P-printable subset, as was to be shown.

(5) The proof of this result is implicit in the most common proof (often credited as Hartmanis's simplification of the proof of Borodin and Demers) of the theorem of Borodin and Demers [BD76], here stated as Theorem 3.1.1, as has been noted independently of the present work by Fenner et al. [FFNR96]. See Selman [Sel88] for related work bearing upon the theorem of Borodin and Demers.

For completeness, we include the proof that NP $\cap$ coNP $\neq$ P implies that P $\not\subseteq \mathrm{EASY}^{\forall}_{\forall}$. Let $L \in$ NP $\cap$ coNP via NPTMs N_L and $N_{\overline{L}}$, that is, $L(N_L) = L$ and $L(N_{\overline{L}}) = \overline{L}$. Assume further that $L \notin$ P.

Consider the following NPTM M. On input x, M nondeterministically guesses whether $x \in L$ or $x \in \overline{L}$, simultaneously guessing certificates (i.e., accepting paths of $N_L(x)$ or $N_{\overline{L}}(x)$) for whichever guess was made. Note that

$$L(M) = L(N_L) \cup L(N_{\overline{L}}) = L \cup \overline{L} = \Sigma^*.$$

Σ^* is a set in P. We claim that, under our assumption that $L \notin$ P, $\Sigma^* \notin \text{EASY}^{\forall}_{\forall}$. Suppose to the contrary that $\Sigma^* \in \text{EASY}^{\forall}_{\forall}$. Then, for the NPTM M accepting Σ^*, there exists an FP function f_M that prints an accepting path of $M(x)$ on each input x. Hence, L can be decided in polynomial time by simply checking which path of the initial branching of $M(x)$ led to acceptance. That is, a DPTM for L, on input x, computes $f_M(x)$ and then checks whether the initial nondeterministic guess of $M(x)$ on the path printed by $f_M(x)$ was either $x \in L$ or $x \in \overline{L}$, and accepts accordingly. This contradicts our assumption that $L \notin$ P. Hence, $\Sigma^* \notin \text{EASY}^{\forall}_{\forall}$. ∎

Finally, we state an interesting observation by Selman. Recall that P $= \text{EASY}^{\forall}_{\forall}$ if and only if $\Sigma^* \in \text{EASY}^{\forall}_{\forall}$. The following claim gives further characterizations of P $= \text{EASY}^{\forall}_{\forall}$ in terms of the question of whether $\text{EASY}^{\forall}_{\forall}$ is closed under complementation.

Claim 3.3.4 [Sel95] *The following three statements are equivalent.*

1. P $= \text{EASY}^{\forall}_{\forall}$.
2. $\text{EASY}^{\forall}_{\forall}$ *is closed under complementation.*
3. *There exists a set* L *in* P *such that* $L \in \text{EASY}^{\forall}_{\forall}$ *and* $\overline{L} \in \text{EASY}^{\forall}_{\forall}$.

Proof. (1) implies (2), and (2) implies (3). To see that (3) implies (1), let L be a set such that $L \in$ P, $L \in \text{EASY}^{\forall}_{\forall}$, and $\overline{L} \in \text{EASY}^{\forall}_{\forall}$. Let M_1 (respectively, M_2) be a DPTM that accepts L (respectively, $\overline{L}$). Let N be an NPTM that accepts Σ^*. Define NPTM N_1 so that on input x, N_1 simultaneously simulates N and M_1, and N_1 accepts if and only if both N and M_1 accept. Observe that every accepting computation of N_1 encodes an accepting computation of N. Similarly, define N_2 to simultaneously simulate N and M_2. Then, $L(N_1) = L$ and $L(N_2) = \overline{L}$. Thus, there exist f_1 and f_2 in FP such that $x \in L$ implies that $f_1(x)$ is an accepting computation of N_1, and $x \in \overline{L}$ implies that $f_2(x)$ is an accepting computation of N_2. Define $f(x) = f_1(x)$ if $x \in L$, and $f(x) = f_2(x)$ if $x \in \overline{L}$. Then, $f \in$ FP and for all x, $f(x)$ contains an encoding of a computation of N on x. Thus, $\Sigma^* \in \text{EASY}^{\forall}_{\forall}$. ∎

Consider the reverse of arrow (10) in Figure 3.2, i.e., the question of whether the statement "P $\not\subseteq \text{EASY}^{\forall}_{\forall}$" implies "$\text{EASY}^{\forall}_{\forall} =$ FINITE." Suppose not. That is, suppose that

P $\neq$ EASY$^{\forall}_{\forall} \neq$ FINITE. Then, there is a set L such that L is infinite, $\overline{L}$ is infinite, $L \in$ P, $L \in$ EASY$^{\forall}_{\forall}$, and $\overline{L} \notin$ EASY$^{\forall}_{\forall}$. In Corollary 3.4.2 below, we will give an oracle relative to which P $\neq$ EASY$^{\forall}_{\forall} \neq$ FINITE. Since Claim 3.3.4 and the above comments relativize, in this world, such a set L indeed exists.

3.4 Negative Results

In this section, we show that some of the results from the previous section are optimal with respect to relativizable techniques. That is, for some of the implications displayed in Figure 3.2, we construct an oracle relative to which the reverse of that implication does not hold. For instance, from part 2 and part 5 of Theorem 3.3.3 and the trivial facts that are shown as arrows (1a) and (1c) in Figure 3.2, we have the following implication chains:

1. If NP $\cap$ coNP is P-bi-immune, then NP is P-immune, which in turn implies that EASY$^{\forall}_{\text{io}} \neq$ NP, and

2. If NP $\cap$ coNP is P-bi-immune, then NP $\cap$ coNP $\neq$ P, which in turn implies that P $\not\subseteq$ EASY$^{\forall}_{\forall}$.

We will prove that the reverse of these chains fails to hold in some relativized world. Even worse, this relativized failure can be shown via proving that not even the trivial parts of the chains are invertible for all oracles. For both chains, this result can be achieved via one and the same oracle, A, to be constructed in the proof of Theorem 3.4.1 below. Moreover, relative to A, the inequalities FINITE $\neq$ EASY$^{\forall}_{\forall} \neq$ P $\neq$ NP can be shown to hold simultaneously, see Corollary 3.4.2.

The main technical contribution in the proof of Theorem 3.4.1 is that we give a novel application of the classic "wide spacing" oracle construction technique: We show that this technique *instantly* yields the non-P-bi-immunity of NP relative to some oracle. The use of the wide spacing technique dates so far back that it is hard to know for sure where it was first used. It certainly played an important role in the early paper by Kurtz [Kur83]. See also the very early use of wide gaps to facilitate the brute-force computation of smaller strings employed by Ladner [Lad75], Baker, Gill, and Solovay [BGS75], and Rackoff [Rac82].

Theorem 3.4.1 *There exists an oracle A such that*

(a) NP^A = PSPACE^A,

(b) NP^A *is* P^A*-immune, and*

(c) NP^A *is not* P^A*-bi-immune.*

Proof. The oracle A will be QBF$\oplus B$, where QBF is any fixed PSPACE-complete problem and the set B is constructed in stages, $B = \bigcup_{j\geq 0} B_j$. Define the function t inductively by

$$t(0) = 2 \quad \text{and} \quad t(j) = 2^{2^{2^{2^{t(j-1)}}}} \quad \text{for } j \geq 1.$$

Define the sets T and T_k, $k \geq 0$, by

$$T_k = \Sigma^{t(k)} \text{ for } k \geq 0, \quad \text{and} \quad T = \bigcup_{k\geq 0} T_k.$$

The construction of B will satisfy the following requirement:

$$B \subseteq T \quad \text{and} \quad \|B \cap T_k\| = 1 \text{ for each } k \geq 0. \tag{3.4.2}$$

Fix an enumeration $\{M_i\}_{i\geq 1}$ of all DPOTMs. For each $i \geq 1$, let p_i be a fixed polynomial bounding the runtime of machine M_j. Without loss of generality, assume that our enumeration satisfies for all $j \geq 1$ that

$$\sum_{i=1}^{\lceil \log j \rceil} p_i(0^{t(j)}) < 2^{t(j)-1}. \tag{3.4.3}$$

Note that this can indeed be assumed, w.l.o.g., by clocking the machines with appropriately slow clocks as is standard. At stage j of the construction, machines $M_1, M_2, \ldots, M_{\lceil \log j \rceil}$ will be *active* unless they have already been canceled during earlier stages. Define the language

$$L_B = \{0^n \mid B \cap \Sigma^{n-1}0 \neq \emptyset\}.$$

Note that L_B is in NP^B, and therefore in NP^A. Let B_{j-1} be the content of B prior to stage j. Initially, let B_0 be the empty set. Stage $j > 0$ of the construction of B is as follows.

Stage j.

Case 1: For no active machine M_i does $M_i^{\text{QBF}\oplus B_{j-1}}(0^{t(j)})$ accept.
Choose the smallest string $w_j \in \Sigma^{t(j)-1}0$ such that w_j is not queried in the computation of $M_i^{\text{QBF}\oplus B_{j-1}}(0^{t(j)})$ for any active machine M_i. Set $B_j := B_{j-1} \cup \{w_j\}$.

Case 2: There exists an active machine M_i such that $M_i^{\text{QBF}\oplus B_{j-1}}(0^{t(j)})$ accepts.
Let $\tilde{i}$ be the smallest such i. Mark machine $M_{\tilde{i}}$ as canceled. Set $B_j := B_{j-1} \cup \{1^{t(j)}\}$.

End of Stage j.

By (3.4.3) above, the sum of the runtimes of all machines that can be active at stage j and run on input $0^{t(j)}$ is strictly less than $2^{t(j)-1}$. Thus, the string w_j, if needed in stage j, indeed exists. In addition, (3.4.3) combined with the widely spaced gaps between the lengths of the strings considered in subsequent stages guarantees that the single stages of the construction do not interfere with each other. To see this, note that for each stage j, no machine that is active at this stage can reach (and query) any string of length $t(j+1)$, that is, no oracle extension at later stages can affect the computations performed in stage j.

Note also that each Case 2 cancels a machine, but for each j, at most $\lceil \log j \rceil$ machines are active in stage j, so Case 2 can happen at most $\lceil \log j \rceil$ times. Hence, Case 1 happens infinitely often in this construction. If Case 1 occurs in some stage, say j', a string from $\Sigma^{t(j')-1}0$ is put into $B_{j'} \subseteq B$ that is strictly larger than any string in $B_{j'-1}$, the initial segment of B constructed in earlier stages. It follows that L_B is an infinite set in NP^A. It remains to prove that (a) NP^A = PSPACE^A, (b) NP^A is P^A-immune, and (c) NP^A is not P^A-bi-immune.

Statement (a) follows immediately from the form of the oracle $A = \text{QBF} \oplus B$ and the fact that QBF is PSPACE-complete.

To prove statement (b), note that each DPOTM M_i is either canceled eventually, or M_i is never canceled. If M_i is canceled, then we have by construction that $0^{t(j)} \in L(M_i^A)$ for some j, yet $0^{t(j)} \notin L_B$, since $B \cap \Sigma^{t(j)-1}0 = \emptyset$. Thus, the language accepted by M_i relative to A, $L(M_i^A)$, is not a subset of L_B. In the other case (i.e., if M_i never is canceled), we will argue that $L(M_i^A) \cap L_B$ must be a finite set. Indeed, let s_i be the first stage in which all machines M_ℓ, with $\ell < i$, that will ever be canceled are already canceled. Then, for no stage j with $j \geq s_i$ will $M_i^{\text{QBF}\oplus B_{j-1}}$ accept the input $0^{t(j)}$, as otherwise M_i would have been the first (i.e., having the smallest number in the enumeration) active machine accepting $0^{t(j)}$ and would thus have been canceled. It follows that M_i^A accepts at most a finite number (more precisely, at most $s_i - 1$) of the elements of L_B. To summarize, we have shown that there exists an infinite set in NP^A (namely, L_B) having no infinite subset in P^A, that is, NP^A is P^A-immune.

Finally, we prove statement (c). Suppose there exists an infinite set L in NP^A. Define the function r inductively by

$$r(0) = 2^{2^2} \quad \text{and} \quad r(j) = 2^{2^{2^{r(j-1)}}} \quad \text{for } j \geq 1.$$

Define the sets

$$L_{\text{in}} = \{0^n \mid (\exists j \geq 0)\,[r(j) = n \wedge 0^n \in L]\}$$

and

$$L_{\text{out}} = \{0^n \mid (\exists j \geq 0)\,[r(j) = n \wedge 0^n \notin L]\}.$$

Note that $L_{\text{in}} \subseteq L$ and $L_{\text{out}} \subseteq \overline{L}$, and it holds that either $0^{r(j)} \in L$ for infinitely many j, or $0^{r(j)} \notin L$ for infinitely many j, or both. Thus, either L_{in} is an infinite subset of L, or L_{out} is an infinite subset of $\overline{L}$, or both.

Now we prove that both L_{in} and L_{out} are in P^A. Recall that $A = \text{QBF} \oplus B$ and that $B \subseteq T$ and $\|B \cap T_k\| = 1$, since the construction of B satisfies requirement (3.4.2) above. We now describe a DPOTM M_{in} for which $L(M_{\text{in}}^A) = L_{\text{in}}$. On input x, M_{in}^A first checks whether x is of the form $0^{r(j)}$ for some j. If not, M_{in}^A rejects x. Otherwise, assume $x = 0^{r(k)}$ for some fixed $k \in \mathbb{N}$. In this case, M_{in}^A constructs a potential query table, Q, of all strings in B that can be touched in the computation of the NP^A machine accepting L. Note that all strings in B that are smaller than $|x| = r(k)$ can be found by brute force, since—by definition of the functions r and t—they have lengths at least double-exponentially smaller than $|x|$. For the same reason, no string in B of length not smaller than $|x|$ can be touched in the run of the NP^A machine accepting L, on input x, more than finitely often. All those strings of $B \cap \bigcup_{j \geq k} T_j$ that indeed are queried in this computation can thus be hard-coded into table Q. It follows that Q contains all information of B that can affect the computation of the NP^A machine for L on input x. Hence, again employing the PSPACE-completeness of QBF, M_{in}^A can ask the QBF part of its oracle to simulate that computation, using table Q for each query to B. If this simulation returns the answer "$x \in L$," then M_{in} accepts x; otherwise, M_{in} rejects x.

The proof that $L_{\text{out}} \in \text{P}^A$ is analogous and thus omitted.

To summarize, we have shown that if there exists an infinite set L in NP^A, then at least one of L or $\overline{L}$ must contain an infinite subset (specifically, L_{in} or L_{out}) that is decidable in P^A, that is, NP^A is not P^A-bi-immune. ∎

Note that the oracle A constructed in the previous proof is recursive and is "reasonable," since $\text{P} \neq \text{NP}$ holds relative to A, due to the P^A-immunity of NP^A. In addition, relative to A, the reverse of arrows (1a) and (1c) in Figure 3.2 fails and $\text{FINITE} \neq \text{EASY}_{\forall}^{\forall} \neq \text{P}$, i.e., arrow (10) in Figure 3.2 is not invertible. This observation is stated in the following corollary.

Corollary 3.4.2 *There exists an oracle A such that the following four statements hold simultaneously.*

1. NP^A *is* P^A*-immune, yet* $\text{NP}^A \cap \text{coNP}^A$ *is not* P^A*-bi-immune.*
2. $\text{NP}^A \cap \text{coNP}^A \neq \text{P}^A$*, yet* $\text{NP}^A \cap \text{coNP}^A$ *is not* P^A*-bi-immune.*
3. $\text{P}^A \not\subseteq (\text{EASY}_{\forall}^{\forall})^A$.
4. $(\text{EASY}_{\forall}^{\forall})^A \cap \text{P}_{\infty}^A \neq \emptyset$.

Proof. Let $A = \text{QBF} \oplus B$ be the oracle constructed in the proof of Theorem 3.4.1. We now argue that this oracle satisfies statements (1) through (4).

The first two statements follow immediately from Theorem 3.4.1. In particular, that NP^A is not P^A-bi-immune implies that $\text{NP}^A \cap \text{coNP}^A$ is not P^A-bi-immune. Moreover, since $\text{NP}^A = \text{PSPACE}^A$ implies that $\text{NP}^A = \text{coNP}^A$, there exists a set in $(\text{NP}^A \cap \text{coNP}^A) - \text{P}^A$ by the P^A-immunity of NP^A.

The relativized version of part 5 of Theorem 3.3.3 establishes the third statement of this corollary: Since $\text{NP}^A \cap \text{coNP}^A \neq \text{P}^A$, we have $\text{P}^A \not\subseteq (\text{EASY}^{\forall}_{\forall})^A$.

To prove the fourth statement, let r and t be the functions and let T_k, $k \geq 0$, and T be the sets defined in the proof of Theorem 3.4.1. Recall that the construction of B ensures that the requirement (3.4.2) is satisfied: $B \subseteq T$ and $\|B \cap T_k\| = 1$ for every $k \geq 0$.

Define the set $L = \{0^{r(j)} \mid j \geq 0\}$. Note that L is an infinite set in P, and thus in P^A.

We now show that $L \in (\text{EASY}^{\forall}_{\forall})^A$. Let N be any NPOTM that, with oracle QBF $\oplus B$, accepts L, i.e., $L(N^{\text{QBF}\oplus B}) = L$. To show that N, with oracle QBF $\oplus B$, has always easy certificates, we will describe a DPOTM M that computes, using oracle QBF $\oplus B$, an $\text{FP}^{\text{QBF}\oplus B}$ function f_N such that f_N prints an accepting path of $N^{\text{QBF}\oplus B}(x)$ for each $x \in L$.

On input x of the form $0^{r(j)}$ for some j, M computes by brute force a potential query table, Q, of all "short" strings in B, i.e., $Q = B^{<r(j)} = B^{\leq t(j)}$, and then employs the PSPACE-completeness of QBF to construct, bit by bit, the lexicographically first accepting path of $N^{\text{QBF}\oplus B}(x)$, say p, via prefix search, where table Q is used to answer all queries to B. Since by definition of r and t, $|x| = r(j)$ is at least double-exponentially smaller than any string of length $> t(j)$, no string in B of length $> t(j)$ can be queried in the run of $N^{\text{QBF}\oplus B}(x)$ more than finitely often. All those strings in $B \cap \bigcup_{i>t(j)} T_i$ that indeed are queried in this computation can thus be hard-coded into table Q. It follows that Q contains all information of B that can affect the computation of $N^{\text{QBF}\oplus B}(x)$. Thus, the path p constructed by $M^{\text{QBF}\oplus B}(x)$ indeed is a valid accepting path of $N^{\text{QBF}\oplus B}(x)$. M outputs p.

Hence, $L \in (\text{EASY}^{\forall}_{\forall})^A$. This establishes our claim that $(\text{EASY}^{\forall}_{\forall})^A \cap \text{P}^A_{\infty} \neq \emptyset$. ∎

Remark 3.4.3 *In fact, it is not hard to see that, via using a Kolmogorov complexity based oracle construction, we can even prove the following claim that is stronger than part 3 of Corollary 3.4.2 above: There exists an oracle D such that $\text{P}^D \not\subseteq (\text{EASY}^{\forall}_{\text{io}})^D$.*

To be a bit more precise, the set $L = \{0^{t(j)} \mid j \geq 0\}$ is in P*, and thus in P^X for any X. However, one can construct an oracle set D such that there exists an NPOTM N with $L(N^D) = L$, yet N^D has only hard certificates almost everywhere, i.e., $L \notin (\text{EASY}^{\forall}_{\text{io}})^D$.*

As mentioned earlier, the implication $\text{P} \neq \text{EASY}^{\forall}_{\forall} \implies \text{P} \neq \text{NP}$ (i.e., arrow (6) in Figure 3.2) follows immediately from Theorem 3.3.1. However, Theorem 3.4.4 below states that the reverse of this implication fails in some relativized world.

Theorem 3.4.4 *There exists an oracle A such that* $\mathrm{NP}^A \neq \mathrm{P}^A = (\mathrm{EASY}^{\forall}_{\forall})^A$.

Before we turn to the proof of Theorem 3.4.4, we note that Theorem 3.4.4 improves on a classical result of Baker, Gill, and Solovay [BGS75]: There exists an oracle E relative to which $\mathrm{P}^E \neq \mathrm{NP}^E$, yet $\mathrm{P}^E = \mathrm{NP}^E \cap \mathrm{coNP}^E$. This result states that the (trivial) implication $\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP} \implies \mathrm{P} \neq \mathrm{NP}$ is irreversible up to the limits of relativizing techniques. Indeed, that Theorem 3.4.4 is stronger than the theorem of Baker, Gill, and Solovay is due to the theorem of Borodin und Demers: The implication $\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP} \implies \mathrm{P} \neq \mathrm{NP}$ can be partitioned into

$$\left(\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP} \implies \mathrm{P} \neq \mathrm{EASY}^{\forall}_{\forall}\right) \wedge \left(\mathrm{P} \neq \mathrm{EASY}^{\forall}_{\forall} \implies \mathrm{P} \neq \mathrm{NP}\right), \tag{3.4.4}$$

and both implications in (3.4.4) hold true in every relativization. The two implications in (3.4.4) are shown as respectively arrow (5) and arrow (6) in Figure 3.2.

Corollary 3.4.5 (cf. [BGS75]) *The oracle A to be constructed in the upcoming proof of Theorem 3.4.4 satisfies that* $\mathrm{NP}^A \neq \mathrm{P}^A = \mathrm{NP}^A \cap \mathrm{coNP}^A$.

Proof of Corollary 3.4.5. This statement follows from Theorem 3.4.4 and the fact that the proof of part 5 of Theorem 3.3.3 relativizes. That is, stating the contrapositive of part 5 of Theorem 3.3.3: For every oracle B, if $\mathrm{P}^B \subseteq (\mathrm{EASY}^{\forall}_{\forall})^B$, then $\mathrm{P}^B = \mathrm{NP}^B \cap \mathrm{coNP}^B$. ∎

Remark 3.4.6 *Naor and Impagliazzo* [IN88, Proposition 4.2] *(see also the papers* [CS93, FR94, FFNR96]*) prove that in some relativized world, the reverse of the theorem of Borodin and Demers does not hold, that is, none of the two implications in* (3.4.4) *is reversible by relativizing techniques. Hence, their result too improves on the above-mentioned result of Baker, Gill, and Solovay* [BGS75]. *Note that Fenner et al.* [FFNR96] *present a relativization in which even a condition slightly stronger than "*$\mathrm{P} \neq \mathrm{EASY}^{\forall}_{\forall}$*" cannot imply the condition "*$\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP}$*," and thus they have proven a statement slightly stronger than that the reverse of arrow* (5) *fails in this relativized world.*

Now we turn to the proof of Theorem 3.4.4.

Proof of Theorem 3.4.4. This proof is an adaptation of the proof technique that Baker, Gill, and Solovay [BGS75] used to establish their result. As in their proof, we will apply a priority argument in the oracle construction. In what follows, we will recall the crucial ideas of Baker, Gill, and Solovay's oracle construction (see [BGS75, proof of Theorem 6]), pointing out the differences to our construction.

The oracle A will again be QBF $\oplus B$, where QBF is any fixed PSPACE-complete problem. Again, $B = \bigcup_{n \geq 0} B_n$ is constructed in stages, and for every $n > 0$, B_{n-1} denotes the content of B prior to stage n. Initially, set B_0 to the empty set.

As in [BGS75], define the function e by $e(0) = 2$ and $e(j) = 2^{2^{e(j-1)}}$ for $j \geq 1$. At stage n of the construction, at most one string of length $e(n)$ is added to B so as to simultaneously ensure both $\mathrm{P}^A \neq \mathrm{NP}^A$ and $\mathrm{P}^A \subseteq (\mathrm{EASY}^{\forall}_{\forall})^A$.

Fix an enumeration $\{M_j\}_{j\geq 1}$ of all DPOTMs and an enumeration $\{N_j\}_{j\geq 1}$ of all NPOTMs. For each $j \geq 1$, let p_j be a fixed polynomial bounding the runtime of both M_j and N_j. Define the language

$$L_B = \{0^n \mid (\exists w)\,[w \in B^{=n}]\}.$$

Note that L_B is in NP^B, and therefore in NP^A.

There are two types of requirements to be satisfied in the construction of B. Intuitively, satisfying a requirement of the form $\langle i, i\rangle$ will ensure that $L(M_i^A) \neq L_B$. Thus, satisfying $\langle i, i\rangle$ for each $i \geq 1$ will establish our claim that $\mathrm{P}^A \neq \mathrm{NP}^A$. On the other hand, satisfying a requirement of the form $\langle j, k\rangle$ with $j \neq k$ will ensure that $L(N_k^A) \neq L(M_j^A)$, and N_k^A is thus not a machine accepting the P^A set $L(M_j^A)$. That is, that N_k^A is a machine accepting $L(M_j^A)$ can happen only if requirement $\langle j, k\rangle$ with $j \neq k$ is never satisfied. Thus, to establish our claim that $\mathrm{P}^A \subseteq (\mathrm{EASY}^{\forall}_{\forall})^A$, it suffices to show that N_k^A has always easy certificates for every requirement $\langle j, k\rangle$ with $j \neq k$ that is *never* satisfied.

In more detail, an unsatisfied requirement $\langle i, i\rangle$ is *vulnerable* at stage n of the construction of B if $p_i(e(n)) < 2^{e(n)}$. An unsatisfied requirement $\langle j, k\rangle$ with $j \neq k$ is *vulnerable* at stage n if there exists a string x such that

$$e(n-1) < \log|x| \leq e(n) \leq \max\{p_j(|x|), p_k(|x|)\} < e(n+1) \tag{3.4.5}$$

and in addition it holds that $x \in L(M_j^{\mathrm{QBF}\oplus B_{n-1}})$ if and only if $x \notin L(N_k^{\mathrm{QBF}\oplus B_{n-1}})$. Note that the definition of vulnerability for this second type of an unsatisfied requirement is different from that given in the proof of [BGS75, Theorem 6]. By convention, we agree that requirement R_1 has higher priority than requirement R_2 exactly if $R_1 < R_2$. Stage $n > 0$ of the construction of B is as follows.

Stage n. The requirement of highest priority that is vulnerable at stage n will be satisfied. To satisfy requirement $\langle j, k\rangle$ with $j \neq k$, we simply add no string to B in this stage, i.e., $B_n := B_{n-1}$. To satisfy requirement $\langle i, i\rangle$, simulate the computation of $M_i^{\mathrm{QBF}\oplus B_{n-1}}(0^{e(n)})$. If it rejects, then let w_n be the smallest string of length $e(n)$ that is not queried along the computation of $M_i^{\mathrm{QBF}\oplus B_{n-1}}(0^{e(n)})$, and set $B_n := B_{n-1} \cup \{w_n\}$. If $M_i^{\mathrm{QBF}\oplus B_{n-1}}$ accepts $0^{e(n)}$, then set $B_n := B_{n-1}$.

End of Stage n.

Each requirement $\langle i, i\rangle$ is eventually satisfied, since there are only finitely many requirements of higher priority. Suppose requirement $\langle i, i\rangle$ is satisfied at stage n. Then, since $\langle i, i\rangle$ is vulnerable at stage n, we have $p_i(e(n)) < 2^{e(n)}$. This implies that the

string w_n, if needed to be added to B in stage n, must exist, and further that no string in B of length $> e(n)$ can be touched in the run of $M_i^A(0^{e(n)})$. Since by construction also w_n is not queried by $M_i^A(0^{e(n)})$, we conclude that oracle extensions at stages $\geq n$ do not affect the computation of $M_i^{\mathrm{QBF}\oplus B_{n-1}}(0^{e(n)})$. Hence, $0^{e(n)} \notin L(M_i^{\mathrm{QBF}\oplus B})$ if and only if $0^{e(n)} \notin L(M_i^{\mathrm{QBF}\oplus B_{n-1}})$. By construction, $0^{e(n)} \notin L(M_i^{\mathrm{QBF}\oplus B_{n-1}})$ if and only if there exists some string $w_n \in B_n \cap \Sigma^{e(n)} = B^{=e(n)}$. By definition of L_B, there exists some string $w_n \in B^{=e(n)}$ if and only if $0^{e(n)} \in L_B$. It follows that $L(M_i^{\mathrm{QBF}\oplus B}) \neq L_B$. Since each requirement $\langle i, i\rangle$ is eventually satisfied, we have $L(M_i^{\mathrm{QBF}\oplus B}) \neq L_B$ for each i; so $L_B \in \mathrm{NP}^A - \mathrm{P}^A$.

It remains to prove that $\mathrm{P}^A \subseteq (\mathrm{EASY}_\forall^\forall)^A$. The remainder of this proof is different from the proof of [BGS75, Theorem 6]. Given any pair of machines, M_j and N_k with $j \neq k$, we will show that either $L(N_k^A) \neq L(M_j^A)$, or N_k^A has always easy certificates, thus proving that for every set in P^A, each NP^A machine accepting it has always easy certificates, in symbols, $\mathrm{P}^A \subseteq (\mathrm{EASY}_\forall^\forall)^A$.

Each requirement $\langle j, k\rangle$ with $j \neq k$ is either satisfied eventually, or $\langle j, k\rangle$ is never satisfied. Suppose requirement $\langle j, k\rangle$ is satisfied at stage n for some n. Then, by the definition of vulnerability for this type of requirements, there exists a string x such that (i) $x \in L(M_j^{\mathrm{QBF}\oplus B_{n-1}})$ if and only if $x \notin L(N_k^{\mathrm{QBF}\oplus B_{n-1}})$, (ii) $B_n = B_{n-1}$, and (iii) neither M_j nor N_k, on input x, can query any string of length exceeding $e(n)$. Thus, $x \in L(M_j^A)$ if and only if $x \notin L(N_k^A)$, i.e., N_k^A cannot accept the P^A set $L(M_j^A)$.

So suppose requirement $\langle j, k\rangle$ is never satisfied, i.e., $L(N_k^A) = L(M_j^A)$ might now happen. In this case, it suffices to show that $L(N_k^A) = L(M_j^A)$ implies that N_k^A has always easy certificates. Since this holds for all k for which N_k^A can accept $L(M_j^A)$, we have $L(M_j^A) \in (\mathrm{EASY}_\forall^\forall)^A$.

Let $s_{j,k}$ be the first stage such that (a) and (b) hold:

(a) For every x with $|x| \geq e(s_{j,k})$ there is at most one n such that

$$\log|x| \leq e(n) \leq \max\{p_j(|x|), p_k(|x|)\}.$$

(b) All requirements of higher priority than $\langle j, k\rangle$ that will ever be satisfied are already satisfied.

We will now show that N_k^A on input x has easy certificates for every string x accepted by M_j^A. We describe an FP^A function f_k that, on input x, uses oracle $A = \mathrm{QBF} \oplus B$ to output some accepting path of $N_k^A(x)$ if $x \in L(M_j^A)$.

On input x, if $|x| < e(s_{j,k})$, then f_k uses a finite table to find and output some accepting path of $N_k^A(x)$ whenever $x \in L(M_j^A)$. Otherwise (i.e., if $|x| \geq e(s_{j,k})$), f_k calculates the smallest n such that $e(n) \geq \log|x|$. Then, f_k builds a table, T, of all strings that were

added to B before stage n, i.e., $T = B^{<e(n)}$, by querying its oracle B about *all* strings of lengths $e(0), e(1), \ldots, e(n-1)$. Since $e(n-1) < \log |x|$, only $\mathcal{O}(|x|)$ queries are required in this brute-force search. We have to consider two cases.

Case 1: $e(n) > \max\{p_j(|x|), p_k(|x|)\}$. Then, neither $M_j^A(x)$ nor $N_k^A(x)$ can query their oracle about any string of length $\geq e(n)$. Therefore, the computation of M_j and N_k on input x with oracle QBF $\oplus\, T$ is the same as with oracle QBF $\oplus\, B$. Hence, f_k can run $M_j^{\mathrm{QBF}\oplus T}$ on input x to determine whether M_j^A would accept x. If it rejects x, then f_k can output an arbitrary string, and we are done. If $M_j^{\mathrm{QBF}\oplus T}$ accepts x, then f_k exploits the PSPACE-completeness of QBF to construct the lexicographically first accepting path of $N_k^{\mathrm{QBF}\oplus B}(x)$, say p, bit by bit via prefix search, where QBF uses table T to answer every oracle call of $N_k^{\mathrm{QBF}\oplus B}(x)$ to B. It follows that p is a valid accepting path of $N_k^A(x)$ if $x \in L(M_j^A)$. f_k outputs p.

Case 2: $e(n) \leq \max\{p_j(|x|), p_k(|x|)\}$. In this case, also strings of length $e(n)$ can be queried by $M_j^A(x)$ or $N_k^A(x)$. Note that $N_k^{\mathrm{QBF}\oplus T}$ accepts x if and only if $M_j^{\mathrm{QBF}\oplus T}$ accepts x, as otherwise requirement $\langle j, k\rangle$ would have been satisfied at stage n, contradicting our supposition that $\langle j, k\rangle$ is never satisfied. Indeed, since $\langle j, k\rangle$ is the smallest unsatisfied requirement at stage n by (b) above and since x meets condition (3.4.5) above by (a), the equivalence

$$x \in L(M_j^{\mathrm{QBF}\oplus T}) \iff x \notin L(N_k^{\mathrm{QBF}\oplus T})$$

would enforce the vulnerability of $\langle j, k\rangle$ at this stage. Now, f_k simulates $M_j^A(x)$ and outputs an arbitrary string if it rejects. Otherwise (i.e., if $x \in L(M_j^A)$), f_k runs $M_j^{\mathrm{QBF}\oplus T}(x)$, call this computation q. There are two subcases.

Case 2.1: The computation of $M_j^A(x)$ exactly agrees with q. Then, there exists an accepting path of $N_k^{\mathrm{QBF}\oplus T}(x)$, and f_k again employs QBF to construct the lexicographically first accepting path of $N_k^{\mathrm{QBF}\oplus T}(x)$, call this path p. If p were reliable w.r.t. oracle A, then f_k could simply output p, and we were done. However, p is not reliable, since T and B might differ, so f_k has to check the validity of p. By our choice of $s_{j,k}$, there exists (according to (a) above) at most one n such that

$$\log |x| \leq e(n) \leq \max\{p_j(|x|), p_k(|x|)\}.$$

Hence, T can lack at most one length $e(n)$ string of B that might be queried in the run of $N_k^A(x)$. Now, f_k checks whether p is a valid certificate of $N_k^A(x)$ by simply checking whether all answers given along p are correct according to the B part of f_k's oracle. There are two subcases of Case 2.1.

Case 2.1.1: All strings z queried along p receive the answer "yes" if and only if $z \in B$. We conclude that p is a valid certificate of $N_k^A(x)$. f_k outputs p.

Case 2.1.2: There exists a string z that is queried along p, but receives a wrong "no" answer according to B, i.e., $z \in B$. Then, f_k has detected that z is the one string of length $e(n)$ in $B - T$. So, adding z to T, we now have $T = B^{\leq e(n)}$. f_k again employs QBF to construct the lexicographically first accepting path of $N_k^{\mathrm{QBF}\oplus T}(x)$, say p', which now must be a valid certificate of $N_k^A(x)$. f_k outputs p'.

Case 2.2: The computation of $M_j^A(x)$ differs from q. The only way this could happen, however, is that the one missing string in T, $z \in B^{\leq e(n)} - T$, is queried on q, but has received a wrong "no" answer from T. Then, as in Case 2.1.2, f_k has identified z and can complete table T by adding z to T. Now, $T = B^{\leq e(n)}$, and f_k can proceed as in Case 2.1.2 to find and output a valid certificate of $N_k^A(x)$ if x is accepted by M_j^A (via once more employing QBF in the prefix search to construct the lexicographically first certificate).

Since $\max\{p_j(|x|), p_k(|x|)\} < e(n+1)$ by (a) above, no string of length $\geq e(n+1)$ can be queried by N_k or M_j on input x, and thus oracle extensions at stages $\geq n+1$ cannot affect the computation of $N_k^{\mathrm{QBF}\oplus B_n}(x)$ or $M_j^{\mathrm{QBF}\oplus B_n}(x)$. This completes the proof. ∎

Remark 3.4.7 *The argument given in the above proof could almost seem to even prove that there exists some oracle A relative to which* $\mathrm{P}^A \neq \mathrm{NP}^A$ *and* $\mathrm{NP}^A \subseteq (\mathrm{EASY}_{\forall}^{\forall})^A$*, i.e.,* $\mathrm{P}^A \neq \mathrm{NP}^A$ *and* $\mathrm{P}^A = \mathrm{NP}^A$*, which is impossible. However, our argument does* not *work for* NP *in place of* P*, for the following subtle reason. When we define the vulnerability of requirements of the form $\langle j, k\rangle$ with $j \neq k$ in terms of pairs of two* nondeterministic *oracle machines N_j and N_k, and modify our argument appropriately, then the* FP^A *function f_k has no way of telling whether or not the input x is accepted by N_j^A (since we have no* P^A *algorithm as in the above proof) and therefore is in serious trouble when it is trying to construct a valid certificate of $N_k^A(x)$.*

Of course, the existence of relativized worlds A in which a statement X^A fails should not be viewed as evidence that X fails in the unrelativized world. Rather, the existence of such relativized worlds should be viewed as evidence that most standard proof techniques lack the power to prove that X holds in the unrelativized world. See, e.g., [All90, For94, Har85] for discussions of how to interpret relativized results. We suggest as an open question the issue of whether even stronger implications than those of Figure 3.2 can be established.

As a final remark, an anonymous referee of the journal paper [HRW97a] mentioned the interesting open topic of definitions analogous to ours, except with the path-finding operator being probabilistic (either error-bounded or error-unbounded), rather than deterministic.

Chapter 4

One-Way Permutations and Associative One-Way Functions in Complexity Theory

4.1 Introduction

It is well known that (many-to-one) one-way functions exist if and only if P $\neq$ NP. Thus, we cannot hope for an ultimate solution to the question of whether or not one-way functions exist unless we can solve the famous P $\stackrel{?}{=}$ NP problem. All we can hope for is to characterize the existence of certain special types of one-way functions via complexity-theoretic statements such as the collapse or separation of complexity classes.

Many types of one-way functions have been studied in the literature. Most notable among such results is Grollmann and Selman's characterization of the existence of certain types of *injective* one-way functions by conditions such as P $\neq$ UP $\cap$ coUP or P $\neq$ UP [GS88], see also [Ko85]. Allender [All86] extended their results by proving that *poly-to-one* one-way functions exist if and only if P $\neq$ FewP.[1] Watanabe [Wat88] showed that *constant-to-one* one-way functions exist if and only if injective one-way functions exist. Watanabe [Wat92] also showed that the existence of *randomized*, *injective* one-way functions and the existence of *extensible*, *injective* one-way functions, respectively, can be

[1] UP and FewP are very interesting complexity classes whose important applications include but extend beyond the study of one-way functions. They have been thoroughly studied in a wide variety of topics, including the following: the relative complexity of checking and evaluating [Val76]; machine representations, complete sets, the Isomorphism Conjecture of Berman and Hartmanis, and relativizations [HH88a, HH91, HH90, HR92]; Boolean hierarchy equivalences [HR97b, HR95]; circuit complexity [HR97b]; fault-tolerant data base access [CHV93]; upward separation [All91, RRW94]; and relations to counting classes [CH90, KSTT92, OH93, FFK94].

characterized by the separations BPP $\neq$ UP$^{\text{BPP}}$ and $\mathcal{P} \neq \mathcal{UP}$, where $\mathcal{P}$ is the class of polynomial-time solvable promise problems (in the sense of [EY80, ESY84, GS88]) and $\mathcal{UP}$ is the class of unambiguous promise problems [CHV93, HR97b]. Finally, Fenner et al. [FFNR96] proved the existence of *surjective, many-to-one* one-way functions equivalent to P $\not\subseteq$ EASY$^{\forall}_{\forall}$, using different nomenclature.

In this chapter, we provide characterizations of the existence of associative one-way functions and of partial and total one-way permutations.

Rabi and Sherman [RS97, RS93] study associative one-way functions (AOWFs) and show that AOWFs exist exactly if P $\neq$ NP. They also present the notion of strong AOWFs—AOWFs that are hard to invert even when one of their arguments is given. They give protocols due to Rivest and Sherman for two-party secret-key agreement and due to Rabi and Sherman for digital signatures, that depend on strong, total AOWFs. They also outline a protocol approach for multi-party secret-key agreement that depends on strong, total, commutative AOWFs.

There are two key worries regarding the Rabi-Sherman approach. The first is whether their protocols are secure even if strong, total, commutative AOWFs exist. This worry has two facets. The first facet is that, as they note, like Diffie-Hellman [DH76, DH79] the protocol they describe has no current proof of security (even if the existence of strong, total, commutative AOWFs is given), though Rabi and Sherman give intuitively attractive arguments suggesting the plausibility of security. In particular, they prove that certain direct attacks against their protocols are precluded by the fact that the protocols use strong, total AOWFs as building blocks. The second facet of the first worry is that their definition of strong, total, commutative AOWFs is a worst-case definition, as opposed to the average-case definition one desires for a satisfyingly strong approach to cryptography.

The second worry is that Rabi and Sherman provide no evidence at all that strong, total, commutative AOWFs exist, though they do prove that AOWFs exist if P $\neq$ NP. In Section 4.2 we completely remove that worry by proving that strong, total, commutative AOWFs exist if P $\neq$ NP.

In light of the above-mentioned first worry—and especially its second facet—we note, as did Rabi and Sherman, that the study of AOWFs should be viewed as more of complexity-theoretic interest than of applied cryptographic interest, though it is hoped that AOWFs will in the long term prove, probably in average-case versions, to be of substantial applied cryptographic value.

Phrasing our work in a slightly different but equivalent way, in Section 4.2 we prove that the existence of AOWFs (or, indeed, the existence of *any* one-way function) implies the existence of strong, total, commutative AOWFs. Furthermore, based on Kleene's [Kle52] distinction between *weak* and *complete equality* of partial functions, we give a definition of associativity that, for partial functions, is a more natural analog of the standard total-

function definition than that of Rabi and Sherman, and we show that their and our results hold even under this more natural definition.

In Section 4.3, we turn to the question of whether or not one-way permutations exist. First, let us briefly digress on the question: What makes NP-complete problems intractable? One possible source of their potential intractability is the fact that there are many possible sets of solutions: The search space is exponential so the cardinality of the set of sets of solutions is double-exponential in the input size. Another possible source of NP's complexity is that all solutions—even if there are just a few of them—may be random in the sense of Kolmogorov complexity and thus hard to find. For both reasons one may try to "remove" the difficulty from NP by considering subclasses of NP that, by definition, contain only easy sets with respect to either type of difficulty. NP's subclasses UP and FewP (defined in Chapter 2) both implicitly reduce the richness of the class of potential solutions to $2^{n^{\mathcal{O}(1)}}$. To single out those NP sets that, for all NP machines accepting them, have Kolmogorov-easy solutions for all instances in the set, we defined the class $\mathrm{EASY}^{\forall}_{\forall}$ in Definition 3.2.1 on page 20. Interestingly, both these concepts of easy NP sets, UP and $\mathrm{EASY}^{\forall}_{\forall}$, have their own connection to the invertibility of certain types of one-way functions, as stated in the second paragraph of this section.

In Section 4.3, characterizations of the existence of various types of one-way permutations are given in terms of complexity-theoretic conditions. In particular, we introduce the UP analog $\mathrm{EASY}^{\forall}_{\forall}(\mathrm{UP})$ of $\mathrm{EASY}^{\forall}_{\forall}$, which combines the restriction of unambiguous computation with the constraint required by $\mathrm{EASY}^{\forall}_{\forall}$. Thus, $\mathrm{EASY}^{\forall}_{\forall}(\mathrm{UP})$ *simultaneously* reduces the solution space of NP problems to at most one solution per input and requires that this one solution, if it exists, can be found in polynomial time. We prove that partial one-way permutations exist if and only if $\mathrm{P} \neq \mathrm{EASY}^{\forall}_{\forall}(\mathrm{UP})$. We also establish a condition necessary and sufficient for the existence of total one-way permutations.

In addition, various types of poly-to-one one-way functions are characterized in terms of class separations such as $\mathrm{P} \neq \mathrm{EASY}^{\forall}_{\forall}(\mathrm{FewP})$, where $\mathrm{EASY}^{\forall}_{\forall}(\mathrm{FewP})$ is the FewP analog of $\mathrm{EASY}^{\forall}_{\forall}$. Based on an observation by Selman ([Sel95], see Claim 3.3.4 on page 31), we also show that the existence of one-way permutations and of poly-to-one one-way functions, respectively, can be characterized in terms of closure under complementation of $\mathrm{EASY}^{\forall}_{\forall}(\mathrm{UP})$ and $\mathrm{EASY}^{\forall}_{\forall}(\mathrm{FewP})$. Moreover, we discuss our results with respect to the UP analog of the Borodin-Demers theorem ([BD76], see Theorem 3.1.1 on page 19) and with respect to statements of the seemingly unrelated field of (finite model) logic.

This chapter is organized as follows. Section 4.2 studies associative one-way functions. Section 4.3 deals with one-way permutations. Section 4.4 presents conclusions and describes some open issues.

4.2 Creating Strong, Total, Commutative, Associative One-Way Functions from Any One-Way Function in Complexity Theory

Section 4.2 is organized as follows. Section 4.2.1 provides definitions and other preliminaries. Section 4.2.2 establishes our main result. Section 4.2.3 discusses an issue related to injectivity. Section 4.2.4 proves that if UP $\neq$ NP then a construction of Rabi and Sherman is invalid.

4.2.1 Preliminaries

Throughout Section 4.2, when we use "binary function" we mean "two-argument function." Recall from Chapter 2 that unless explicitly stated as being total, all functions may potentially be partial,[2] i.e., "let σ be any binary function" does not imply that σ will necessarily be total. For any binary function σ, we will interchangeably use prefix and infix notation, i.e., $\sigma(x, y) = x\sigma y$.

For concreteness, we now explicitly define the notion of one-way-ness for *binary* functions. Regarding part 3 of the following definition, we mention that we use the term "one-way function" in the same way Rabi and Sherman [RS97] do, i.e., in the complexity-theoretic (that is, worst-case) sense, and without requiring that the function necessarily be injective.

Definition 4.2.1 *Let $\sigma : \Sigma^* \times \Sigma^* \to \Sigma^*$ be any binary function.*

1. *We say σ is* honest *if and only if there exists some polynomial p such that for every $z \in$ image(σ) there exists a pair $(x, y) \in$ domain(σ) such that $x\sigma y = z$ and $|x| + |y| \leq p(|z|)$.*[3]

2. *We say σ is* FP-invertible *if and only if there exists a total function $g \in$ FP such that for every $z \in$ image(σ), $g(z)$ is some element of $\sigma^{-1}(z) = \{(x, y) \in$ domain$(\sigma) \mid x\sigma y = z\}$.*

3. *We say σ is a* one-way function *if and only if σ is honest, polynomial-time computable, and not* FP*-invertible.*

[2] However, for rhetorical reasons, we may sometimes explicitly mention that a function is partial.

[3] This definition of honesty for binary functions is that of Rabi and Sherman [RS97], and is equivalent to requiring $|\langle x, y\rangle| \leq p(|z|)$, since there exists some polynomial q (that depends on the pairing function chosen) such that for every $x, y \in \Sigma^*$, $|\langle x, y\rangle| \leq q(|x| + |y|)$ and $|x| + |y| \leq q(|\langle x, y\rangle|)$.

Rabi and Sherman [RS97] define a notion of associativity for binary functions as follows. Diskussion 4.2.3 below explains why we use the term "weakly associative" in Definition 4.2.2 to describe their notion.

Definition 4.2.2 *Let* $\circ : \Sigma^* \times \Sigma^* \to \Sigma^*$ *be any binary function. We say* $\circ$ *is* weakly associative *if and only if* $x \circ (y \circ z) = (x \circ y) \circ z$ *holds for all* $x, y, z \in \Sigma^*$ *such that each of* (x, y), (y, z), $(x, y \circ z)$, *and* $(x \circ y, z)$ *is an element of* domain$(\circ)$.

Discussion 4.2.3 *Associativity expresses equality between two functions each of which can be viewed as a 3-ary function that results from a given binary function. Definition 4.2.2 yields a notion of associativity that is not natural for non-total functions, since it does not evaluate as being false "equations" such as* "undefined $= 1010$." *This situation can occur in* $x \circ (y \circ z) = (x \circ y) \circ z$ *in various ways, e.g., if* (x, y), $(x \circ y, z)$, *and* (y, z) *are in the domain of* $\circ$ *but* $(x, y \circ z)$ *is not. It would seem more natural for a definition of associativity for binary functions to require that both sides of the above equation stand or fall together. That is, for each triple of strings* $x, y, z \in \Sigma^*$*, either both sides should be defined and equal, or each side should be undefined.*

This observation is not new. Drawing on Kleene's careful discussion of how to define equality between partial functions, our definition of associativity—given in Definition 4.2.4 below—achieves this natural behavior. The distinction in the two definitions of associativity can be said to come from two distinct interpretations of "equality" between functions, known in recursive function theory as *weak equality* and *complete equality*, see Kleene [Kle52]. Kleene suggests the use of two different equality symbols. We will use "$=_w$" and "$=_c$," and we have modified the following quotation to use these also. Kleene writes [Kle52, pp. 327–328]:

> We now introduce "$\psi(x_1, \ldots, x_n) =_c \chi(x_1, \ldots, x_n)$" to express, for particular $x_1, \ldots, x_n$, that if either of $\psi(x_1, \ldots, x_n)$ and $\chi(x_1, \ldots, x_n)$ is defined, so is the other and the values are the same (and hence if either of $\psi(x_1, \ldots, x_n)$ and $\chi(x_1, \ldots, x_n)$ is undefined, so is the other). The difference in the meaning of (i) "$\psi(x_1, \ldots, x_n) =_w \chi(x_1, \ldots, x_n)$" and (ii) "$\psi(x_1, \ldots, x_n) =_c \chi(x_1, \ldots, x_n)$" comes when one of $\psi(x_1, \ldots, x_n)$ and $\chi(x_1, \ldots, x_n)$ is undefined. Then (i) is undefined, while (ii) is true or false according as the other is or is not undefined.

Complete equality is the more natural of the two notions, and Definition 4.2.4 yields a notion of associativity for binary functions that is based on complete equality. Nonetheless, we will show that Rabi and Sherman's results [RS97] and our results hold even under this more restrictive definition. In a similar vein, we also define commutativity for (partial) binary functions.

Definition 4.2.4 *Let $\sigma : \Sigma^* \times \Sigma^* \to \Sigma^*$ be any binary function. Define $\Gamma = \Sigma^* \cup \{\perp\}$ and define an extension $\widehat{\sigma} : \Gamma \times \Gamma \to \Gamma$ of σ as follows:*

$$\widehat{\sigma}(a,b) = \begin{cases} \sigma(a,b) & \text{if } a \neq \perp \text{ and } b \neq \perp \text{ and } (a,b) \in \text{domain}(\sigma) \\ \perp & \text{otherwise.} \end{cases} \tag{4.2.1}$$

We say σ is associative *if and only if, for every $x, y, z \in \Sigma^*$, $(x\widehat{\sigma}y)\widehat{\sigma}z = x\widehat{\sigma}(y\widehat{\sigma}z)$. We say σ is* commutative *if and only if, for every $x, y \in \Sigma^*$, $x\widehat{\sigma}y = y\widehat{\sigma}x$ (i.e., $x\sigma y =_c y\sigma x$).*

Every associative function is weakly associative; the converse, however, is not always true, so these are indeed different notions.

Proposition 4.2.5 *The following three statements are true.*

1. *Every associative binary function is weakly associative.*
2. *Every total binary function is associative if and only if it is weakly associative.*
3. *There exists a binary function that is weakly associative, but not associative.*

Proof. (1) and (2) are immediate from the definitions. Regarding (3), note that the following binary function $\sigma : \Sigma^* \times \Sigma^* \to \Sigma^*$ is weakly associative but not associative:

$$\sigma(a,b) = \begin{cases} 111 & \text{if } a = 1 \text{ and } b = 11 \\ 0 & \text{if } a = 111 \text{ and } b = 1111 \\ \text{undefined} & \text{otherwise,} \end{cases} \tag{4.2.2}$$

where by "undefined" above we do not mean some new token "undefined," but rather we simply mean that for cases handled by that line of the definition $(a,b) \notin \text{domain}(\sigma)$. ∎

Definition 4.2.6 1. *A binary function $\sigma : \Sigma^* \times \Sigma^* \to \Sigma^*$ is an* AOWF *if and only if σ is both associative and a one-way function.*

2. [RS97] *A binary function $\sigma : \Sigma^* \times \Sigma^* \to \Sigma^*$ is an* AWOWF *if and only if σ is both weakly associative and a one-way function.*

Rabi and Sherman [RS97] also introduce the notion of *strong* one-way functions—binary one-way functions that are hard to invert even if one of their arguments is given. Strongness implies one-way-ness. (We note that "strongness" here should not be confused with the property of strong-FP-invertibility of functions introduced by Allender [All86, All85].) To avoid any possibility of ambiguity we henceforward, when using equality signs with partial functions, will make it explicit that by equality we mean $=_c$.

Definition 4.2.7 *A binary function σ is said to be* strong *if and only if σ is not* FP*-invertible even if one of its arguments is given. More formally, binary function σ is* strong *if and only if neither (a) nor (b) holds:*

(a) *There exists a total function $g_1 \in$ FP such that for every $z \in$ image(σ) and for each $x \in \Sigma^*$, if $\sigma(x, y) =_c z$ for some $y \in \Sigma^*$, then $\sigma(x, g_1(\langle x, z\rangle)) =_c z$.*

(b) *There exists a total function $g_2 \in$ FP such that for every $z \in$ image(σ) and for each $y \in \Sigma^*$, if $\sigma(x, y) =_c z$ for some $x \in \Sigma^*$, then $\sigma(g_2(\langle y, z\rangle), y) =_c z$.*

4.2.2 Strong, Total, Commutative, Associative One-Way Functions

This section proves that P $\neq$ NP if and only if strong, total, commutative, associative one-way functions exist.

Recall that Rabi and Sherman [RS97] show that AᵂOWFs exist if and only if P $\neq$ NP. However, they present no evidence that *strong* AᵂOWFs exist, and they establish no structural conditions sufficient to imply that any exist. Solving these open questions, we show in Theorem 4.2.8 below that there exist strong, total, commutative AᵂOWFs (equivalently, strong, total, commutative AOWFs) if and only if P $\neq$ NP.

Theorem 4.2.8 *The following five statements are equivalent.*

1. P $\neq$ NP.
2. *There exist AᵂOWFs.*
3. *There exist AOWFs.*
4. *There exist strong, total, commutative AᵂOWFs.*
5. *There exist strong, total, commutative AOWFs.*

Proof. By part 2 of Proposition 4.2.5, (4) and (5) are equivalent. By part 1 of Proposition 4.2.5, (3) implies (2). Rabi and Sherman [RS97] have shown the equivalence of (1) and (2), by exploiting the associativity of the closest common ancestor relation for configurations in the computation tree of nondeterministic Turing machines. (5) (and, equivalently, (4)) implies (2) and (3). So to establish the theorem it suffices to show that (1) implies (5).

We will soon define a key function, σ. We at that point describe the intuition behind it, and we describe the two-phase strategy our proof will follow.

Assume P $\neq$ NP, and let A be a set in NP$-$P. Let M be a nondeterministic polynomial-time Turing machine accepting A. By a *witness* for "$x \in A$" we mean a string $w \in \Sigma^*$

that encodes some accepting computation path of M on input x. Recall from Chapter 2 that for each string x, the set of witnesses for "$x \in A$" (with respect to M) is denoted by $\mathrm{ACC}_M(x)$. Note that if $x \notin A$ then $\mathrm{ACC}_M(x) = \emptyset$. Assume, without loss of generality, that for each $x \in A$, every witness $w \in \mathrm{ACC}_M(x)$ satisfies $|w| = p(|x|) > |x|$ for some strictly increasing polynomial p depending on M.

For any strings $u, v, w \in \Sigma^*$, $\min(u, v)$ will denote the lexicographically smaller of u and v, and $\min(u, v, w)$ will denote the lexicographically smallest of u, v, and w.

Define the binary function $\sigma : \Sigma^* \times \Sigma^* \to \Sigma^*$ by

(4.2.3)

$$\sigma(a,b) = \begin{cases} \langle x, \min(w_1, w_2)\rangle & \text{if } (\exists x \in \Sigma^*)\,(\exists w_1, w_2 \in \mathrm{ACC}_M(x)) \\ & \qquad [a = \langle x, w_1\rangle \ \wedge\ b = \langle x, w_2\rangle] \\ \langle x, x\rangle & \text{if } (\exists x \in \Sigma^*)\,(\exists w \in \mathrm{ACC}_M(x))\,[(a = \langle x, x\rangle \wedge \\ & \qquad b = \langle x, w\rangle) \ \vee\ (a = \langle x, w\rangle \ \wedge\ b = \langle x, x\rangle)] \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Very informally put, the intuition behind σ is that it reduces the number of witnesses by one, in a particular, careful way. "Case 1" below describes this more specifically. Also very informally put, the intuition behind why σ will prove to be hard to invert is that inversion requires obtaining witness information.

Our proof takes a two-step approach. In particular, on our way towards a proof that (1) implies (5), we will first prove that:

Claim A The function σ defined above is a strong, commutative AOWF.

Then we will:

Task B Show how to extend σ to a strong, total, commutative AOWF,

thus establishing (5).

We start on our proof of Claim A. Note that σ is honest. Also, $\sigma \in$ FP. That is, given $\langle a, b\rangle$ as the input, it is easy to decide in polynomial time whether $(a, b) \in \mathrm{domain}(\sigma)$, and if so, which of $\langle x, x\rangle$ or $\langle x, w\rangle$ for suitable $x \in \Sigma^*$ and $w \in \mathrm{ACC}_M(x)$ should be output as the value of $\sigma(a, b)$. Here, we need the assumption that for each $x \in A$, every witness w for "$x \in A$" satisfies $|w| = p(|x|) > |x|$. This assumption ensures that there is no ambiguity in determining whether a and b are of the form $\langle x, x\rangle$ or of the form $\langle x, \text{PotentialWitness}\rangle$, and checking items of the form $\langle x, \text{PotentialWitness}\rangle$ is easy because $\bigcup_{x \in \Sigma^*} \mathrm{ACC}_M(x)$ is a set in P. That $\bigcup_{x \in \Sigma^*} \mathrm{ACC}_M(x)$ is a set in P also ensures that $\mathrm{domain}(\sigma)$ is a set in P.

Now, we show that σ cannot be inverted in polynomial time, even if one of its arguments is given. Assume, for instance, that there exists a total function $g_2 \in$ FP such that given

any z in the image of σ and any second argument b for which there is some $a \in \Sigma^*$ with $\sigma(a,b) =_c z$, it holds that $\sigma(g_2(\langle b, z\rangle), b) =_c z$. Then, contradicting our assumption that $A \notin$ P, A could be decided in polynomial time as follows:

> On input x, to decide whether or not $x \in A$, compute $g_2(\langle\langle x,x\rangle,\langle x,x\rangle\rangle)$, interpret it as a pair $\langle d,e\rangle$, and accept if and only if $d = x$ and $e \in \mathrm{ACC}_M(x)$.

An analogous proof works for the case of a fixed first argument. Thus, neither (a) nor (b) of Definition 4.2.7 holds, so σ is a strong one-way function.

We now prove that σ is associative. Let $\widehat{\sigma}$ be the extension of σ from Definition 4.2.4. Fix any strings $a = \langle a_1, a_2\rangle$, $b = \langle b_1, b_2\rangle$, and $c = \langle c_1, c_2\rangle$ in Σ^*. Let k equal how many of a_2, b_2, and c_2 are in $\mathrm{ACC}_M(a_1)$. For example, if $a_2 = b_2 = c_2 \in \mathrm{ACC}_M(a_1)$, then $k = 3$. To show that

$$(a\widehat{\sigma}b)\widehat{\sigma}c = a\widehat{\sigma}(b\widehat{\sigma}c) \tag{4.2.4}$$

holds, we distinguish the following two cases.

Case 1: $[a_1 = b_1 = c_1 \ \wedge\ \{a_2, b_2, c_2\} \subseteq \{a_1\} \cup \mathrm{ACC}_M(a_1)]$. As mentioned above, σ (and thus $\widehat{\sigma}$) decreases by one the number of witnesses. In particular, $\widehat{\sigma}$ preserves the lexicographic minimum if both arguments contain witnesses for "$a_1 \in A$." If exactly one of its arguments contains a witness for "$a_1 \in A$," then $\widehat{\sigma}$ outputs $\langle a_1, a_1\rangle$. If neither of its arguments contains a witness for "$a_1 \in A$," then σ is not defined and thus $\widehat{\sigma}$ has the value $\perp$. It follows that:

- If $k \in \{0,1\}$, then $(a\widehat{\sigma}b)\widehat{\sigma}c = \perp = a\widehat{\sigma}(b\widehat{\sigma}c)$.
- If $k = 2$, then $(a\widehat{\sigma}b)\widehat{\sigma}c = \langle a_1, a_1\rangle = a\widehat{\sigma}(b\widehat{\sigma}c)$.
- If $k = 3$, then $(a\widehat{\sigma}b)\widehat{\sigma}c = \langle a_1, \min(a_2, b_2, c_2)\rangle = a\widehat{\sigma}(b\widehat{\sigma}c)$.

In each of these three cases Equation 4.2.4 is satisfied.

Case 2: Not Case 1. Then it holds that either $[a_1 \neq b_1 \ \vee\ a_1 \neq c_1 \ \vee\ b_1 \neq c_1]$, or $[a_1 = b_1 = c_1 \ \wedge\ \{a_2, b_2, c_2\} \not\subseteq \{a_1\} \cup \mathrm{ACC}_M(a_1)]$. In light of the definition of σ, we have in both cases that $(a\widehat{\sigma}b)\widehat{\sigma}c = \perp = a\widehat{\sigma}(b\widehat{\sigma}c)$, and Equation 4.2.4 is satisfied.

Hence, σ is associative. Furthermore, it is easy to see from the definition of σ that σ is commutative. Thus, σ is a strong, commutative AOWF, as claimed earlier. So Claim A is established.

To complete the proof, we now show how to extend σ to a strong, *total*, commutative AOWF.[4] That is, we now turn to Task B. Informally put, we will use an appropriately chosen string to plug the holes in σ. The fact that σ is an AOWF rather than merely an AwOWF helps us avoid the key problem—see Section 4.2.4—in Rabi and Sherman's extension attempt.

Fix any string $x_0 \notin A$ (one must exist, since $A \notin$ P). Let a_0 be the pair $\langle x_0, 1x_0 \rangle$. Note that a_0 is neither of the form $\langle x, x \rangle$ for any $x \in \Sigma^*$, nor of the form $\langle x, w \rangle$ for any $x \in \Sigma^*$ and any witness $w \in \mathrm{ACC}_M(x)$ (because $x_0 \notin A$ and thus does not have any witnesses). Note that, by the definition of σ, for each y, $(a_0, y) \notin$ domain(σ) and $(y, a_0) \notin$ domain(σ). Define the total function $\tau : \Sigma^* \times \Sigma^* \to \Sigma^*$ as follows: Whenever $(a, b) \in$ domain(σ), define $\tau(a, b) = \sigma(a, b)$; otherwise, define $\tau(a, b) = a_0$.

The function τ is a strong, total, commutative AOWF. In particular, τ is honest, since for a_0, which is the only string in the image of τ that is not in the image of σ, it holds that $\tau(a_0, a_0) = a_0$ and $|a_0| + |a_0| \leq 2|a_0|$. Also, $\tau \in$ FP, since $\sigma \in$ FP and domain$(\sigma) \in$ P. That τ is strong follows from the facts that image$(\sigma) \subseteq$ image(τ) and σ is strong. Finally, to see that τ is associative, note that if $a\widehat{\sigma}(b\widehat{\sigma}c) = \perp$ then $a\tau(b\tau c) = a_0$ and otherwise $a\tau(b\tau c) = a\widehat{\sigma}(b\widehat{\sigma}c)$. Similarly, if $(a\widehat{\sigma}b)\widehat{\sigma}c = \perp$ then $(a\tau b)\tau c = a_0$ and otherwise $(a\tau b)\tau c = (a\widehat{\sigma}b)\widehat{\sigma}c$. The associativity of τ now follows easily, given that σ is associative.

The commutativity of τ is immediate from the definition of τ and the commutativity of σ. To see why this holds, recall that our definition of commutativity is based on complete equality, and thus $(a, b) \in$ domain(σ) if and only if $(b, a) \in$ domain(σ). Hence, τ is a strong, total, commutative AOWF. ∎

Rabi and Sherman emphasize the importance of explicitly exhibiting strong, total AwOWFs [RS97], since the cryptographic protocols given in [RS97] rely on their existence. Rabi and Sherman also pose as an open issue the problem of whether a strong, total AwOWF can be constructed from any given one-way function [RS93]. The proof of Theorem 4.2.8 solves both these open issues. Indeed, the function τ defined in the above proof shows how to construct a strong, total, commutative AOWF (equivalently, a strong, total, commutative AwOWF) based on any clocked NP machine accepting a language in NP $-$ P. Similarly, the proof of Theorem 4.2.8 shows how, given (as a program) any one-way function, along with its polynomial runtime and honesty bounds, one can obtain a clocked NP machine accepting a language in NP $-$ P. Thus, as the title of Section 4.2 claims, from any given one-way function one can create a strong, total, commutative AOWF (equivalently, a strong, total, commutative AwOWF).

[4] Rabi and Sherman [RS97] give a construction that they claim lifts any AwOWF whose domain is in P to a total AwOWF. However, it is far from clear that their construction achieves this claim. In fact, Section 4.2.4 shows that any proof that their construction is valid would immediately prove that UP $=$ NP.

The previous paragraph should not be read as suggesting that actually implementing such a transformation, for example in the computer language C, would be the work of just a few minutes, or would result in a very short, simple C program.

4.2.3 Injective, Associative One-Way Functions

We mention briefly the issue of injective (i.e., one-to-one) AOWFs and AwOWFs. Rabi and Sherman give no evidence that injective AwOWFs might exist. In fact, they prove that no total AwOWF can be injective. Thus, in light of part 2 of Proposition 4.2.5, no total AOWF can be injective. However, as Theorem 4.2.9 we show that P $\neq$ UP if and only if injective AwOWFs (and indeed injective AOWFs) exist.

The lack of injectivity for total, commutative AOWFs and AwOWFs comes close to following already just from commutativity. Consider any commutative function σ such that there exist elements a and b with $a \neq b$ and $(a, b) \in$ domain(σ). Then $\sigma(a, b) =_c \sigma(b, a)$, and so σ is not injective. Now let us generalize the notion of injectivity so as to keep the general intuition of its behavior, yet so as to not to clash so strongly with commutativity. Given any binary function $\sigma : \Sigma^* \times \Sigma^* \to \Sigma^*$, we say σ is *unordered-injective* if and only if for all $a, b, c, d \in \Sigma^*$, if $(a, b), (c, d) \in$ domain(σ) and $\sigma(a, b) =_c \sigma(c, d)$, then $\{a, b\} = \{c, d\}$. That is, each element $x =_c \sigma(a, b)$ in the image of σ has at most one unordered pair $\{a, b\}$ (possibly degenerate, i.e., $\{a, a\} = \{a\}$) as its preimage. If σ is commutative, then both orderings of this unordered pair, (a, b) and (b, a), will map to x; if not, one cannot know (i.e., it is possible that $\sigma(a, b) =_c x$ yet for some string $y \neq x$ it holds that $\sigma(b, a) =_c y$).

Theorem 4.2.9 *The following five statements are equivalent.*

1. P $\neq$ UP.
2. *There exist injective AwOWFs.*
3. *There exist injective AOWFs.*
4. *There exist strong, commutative, unordered-injective AwOWFs.*
5. *There exist strong, commutative, unordered-injective AOWFs.*

Proof. That (2) implies (1) follows immediately by standard techniques (those of [GS88], but for functions with two arguments). By part 1 of Proposition 4.2.5, (3) implies (2). That (1), (4), and (5) are pairwise equivalent follows as a corollary from the proof of Theorem 4.2.8; note, crucially, that if the definition of σ given in that proof is based on

some set $A \in \mathrm{UP} - \mathrm{P}$, then σ is unordered-injective, since no string x in A can have more than one witness. So it suffices to prove that (1) implies (3).

Assuming $A \in \mathrm{UP} - \mathrm{P}$, define the language $A' = \{1x \mid x \in A\}$. Note that $A' \in \mathrm{UP} - \mathrm{P}$. Let M be some UP machine accepting A'. Let the polynomial p and, for each x, let the witness sets $\mathrm{ACC}_M(x)$ be defined as in the proof of Theorem 4.2.8 (note that, for each $x \in A'$, $\mathrm{ACC}_M(x)$ now is a singleton). Without loss of generality, assume that for each $x \in A'$, the unique witness w certifying that $x \in A'$ starts with a 1 as its first bit, i.e., $w \in 1\Sigma^*$. Define the binary function $\sigma : \Sigma^* \times \Sigma^* \to \Sigma^*$ as follows:

$$\sigma(a,b) = \begin{cases} 0a & \text{if } a \in A' \text{ and } \mathrm{ACC}_M(a) = \{b\} \\ \text{undefined} & \text{otherwise.} \end{cases} \tag{4.2.5}$$

Let $\widehat{\sigma}$ be the extension of σ as in Definition 4.2.4. Note that for all $a, b, c \in \Sigma^*$, it holds that $(a\widehat{\sigma}b)\widehat{\sigma}c = \perp = a\widehat{\sigma}(b\widehat{\sigma}c)$ by definition of σ. Thus, σ is associative according to Definition 4.2.4. Also, note that σ is injective, and the standard proof approach—see, e.g., the proof of Theorem 4.2.8—shows that σ is a one-way function. ∎

4.2.4 On a Construction of Rabi and Sherman

As mentioned in footnote 4, Rabi and Sherman [RS97] give a construction that they claim lifts any AwOWF whose domain is in P to a total AwOWF. It is far from clear that their construction achieves this claim. In fact, we show that any proof that their construction is valid would immediately prove that UP = NP. In particular, we provide the following counterexample to Rabi and Sherman's assertion.

Theorem 4.2.10 *If* UP $\neq$ NP, *then there exists an* AwOWF $\tilde{\sigma}$, *satisfying* $(\exists \tilde{a})[(\tilde{a}, \tilde{a}) \notin \mathrm{domain}(\tilde{\sigma})]$ *and having domain in* P, *such that the construction that Rabi and Sherman claim converts* AwOWFs *into total* AwOWFs *in fact fails on* $\tilde{\sigma}$.

Proof. The general idea behind this proof is that if UP $\neq$ NP then the Rabi-Sherman construction does not always preserve weak associativity.

Fix a set $A' \in \mathrm{NP} - \mathrm{UP}$ and an NP machine M' accepting A'. Let the polynomial p' and, for each x, let the witness sets $\mathrm{ACC}_{M'}(x)$ be defined analogously to the definitions of p and $\mathrm{ACC}_M(x)$ in the proof of Theorem 4.2.8 above.

Define the binary function $\tilde{\sigma} : \Sigma^* \times \Sigma^* \to \Sigma^*$ by

$$\tilde{\sigma}(a,b) = \begin{cases} \langle x, w\rangle & \text{if } (\exists x \in \Sigma^*)\,(\exists w \in \mathrm{ACC}_{M'}(x))\,[a = \langle x, w\rangle = b] \\ \langle x, x\rangle & \text{if } (\exists x \in \Sigma^*)\,(\exists w \in \mathrm{ACC}_{M'}(x))\,[(a = \langle x, x\rangle \ \wedge\ b = \langle x, w\rangle) \\ & \qquad\qquad \vee\ (a = \langle x, w\rangle \ \wedge\ b = \langle x, x\rangle)] \\ \text{undefined} & \text{otherwise.} \end{cases} \tag{4.2.6}$$

It is not hard to verify that $\tilde{\sigma}$ is an AwOWF.

Let $\tilde{a}$ be a fixed string such that $(\tilde{a}, \tilde{a}) \notin \text{domain}(\tilde{\sigma})$. For the particular function $\tilde{\sigma}$ defined above, such a string $\tilde{a}$ indeed exists (e.g., let $\tilde{a} = \langle x_0, 1x_0 \rangle$ for any particular fixed $x_0 \notin A'$, see the discussion of a_0 in the proof of Theorem 4.2.8 as to why this is right). In contrast, the "c" of [RS97, p. 242, l. 10] may not in general exist.

Now, using the Rabi-Sherman technique, extend $\tilde{\sigma}$ to a total function, $\tilde{\tau}$, the same way we obtained the total extension "τ" of "σ" in the proof of Theorem 4.2.8. Fix some string $\tilde{x} \in A'$ that has two distinct witnesses w_1 and w_2 in $\text{ACC}_{M'}(\tilde{x})$ (such $\tilde{x}$, w_1, and w_2 exist, as $A' \notin$ UP), and let $a = \langle \tilde{x}, w_1 \rangle$, $b = \langle \tilde{x}, w_2 \rangle$, and $c = \langle \tilde{x}, \tilde{x} \rangle$. Then, we have $(a\tilde{\tau}b)\tilde{\tau}c = \tilde{a} \neq \langle \tilde{x}, \tilde{x} \rangle = a\tilde{\tau}(b\tilde{\tau}c)$. Thus $\tilde{\tau}$ is not associative (and thus, as it is total, is not weakly associative).

The reason that $(a\tilde{\tau}b)\tilde{\tau}c = \tilde{a}$ may not be clear to the reader. To see why this holds, one must look at the Rabi-Sherman technique of extending $\tilde{\sigma}$ to $\tilde{\tau}$, which, very informally, is to use $\tilde{a}$ as a dumping ground. We mention that, for essentially the same reason, $\tilde{\sigma}$ is not associative (and thus is not an AOWF), since $(a\widehat{\tilde{\sigma}}b)\widehat{\tilde{\sigma}}c = \perp \neq \langle \tilde{x}, \tilde{x} \rangle = a\widehat{\tilde{\sigma}}(b\widehat{\tilde{\sigma}}c)$, where $\widehat{\tilde{\sigma}}$ is the extension of $\tilde{\sigma}$ from Definition 4.2.4. ∎

Even if Rabi and Sherman's proof were valid, their claim would not be particularly useful to them, as the AwOWFs they construct [RS97, proof of Theorem 5] do not in general have domains that are in P. In contrast, the function σ of our proof of Theorem 4.2.8 *does* have a domain that is in P, and their method (corrected to remove the "c" problem) does preserve associativity (note: we did not say weak associativity), and so proved useful to us.

4.3 Characterizations of the Existence of Partial and Total One-Way Permutations

Section 4.3 is organized as follows. Section 4.3.1 defines the classes $\text{EASY}^{\forall}_{\forall}(\text{UP})$ and $\text{EASY}^{\forall}_{\forall}(\text{FewP})$, the UP and FewP analogs of $\text{EASY}^{\forall}_{\forall}$, and characterizes the existence of partial one-way permutations and of poly-to-one one-way functions in terms of separations between P and those two classes. Section 4.3.2 provides a condition necessary and sufficient for the existence of total one-way permutations.

4.3.1 Partial One-Way Permutations and Poly-to-One One-Way Functions

Recall from Definition 3.2.1 on page 20 that $\text{EASY}^{\forall}_{\forall}$ is the class of all sets L for which all NPTMs accepting L always (i.e., on all inputs $x \in L$) have easy certificates. We now

define the UP and FewP analogs of $\mathrm{EASY}_{\forall}^{\forall}$. (Though it is clear that a more general definition of the form $\mathrm{EASY}_{\forall}^{\forall}(\mathcal{C}, \mathcal{F})$ for complexity classes $\mathcal{C}$ other than NP, UP, or FewP and for function classes $\mathcal{F}$ other than FP can analogously be obtained, we will only define the classes of interest here.)

Definition 4.3.1 *For $\mathcal{C} \in \{\mathrm{NP}, \mathrm{UP}, \mathrm{FewP}\}$, define $\mathrm{EASY}_{\forall}^{\forall}(\mathcal{C})$ to be the class of all sets L that either are finite, or that satisfy (a) $L \in \mathcal{C}$, and (b) for every $\mathcal{C}$ machine N such that $L(N) = L$, there exists an FP function f_N such that, for all $x \in L$, $f_N(x) \in \mathrm{ACC}_N(x)$.*

The inclusions summarized in Proposition 4.3.2 below follow immediately from the definition. For instance, the first inclusion holds by the following arguments: (a) each $\mathrm{EASY}_{\forall}^{\forall}$ set L is in P and thus in FewP, see Figure 3.1 on page 22; and (b) if every NPTM accepting L always has easy certificates, then so does every FewP machine. Hence $\mathrm{EASY}_{\forall}^{\forall} \subseteq \mathrm{EASY}_{\forall}^{\forall}(\mathrm{FewP})$. The inclusion $\mathrm{EASY}_{\forall}^{\forall}(\mathrm{UP}) \subseteq \mathrm{P}$ holds due to $\mathrm{EASY}_{\forall}^{\forall}(\mathrm{UP}) \subseteq \mathrm{EASY}_{\forall}^{\exists}(\mathrm{UP}) = \mathrm{EASY}_{\forall}^{\exists} = \mathrm{P}$ (see Theorem 3.2.4 for the final equality), where $\mathrm{EASY}_{\forall}^{\exists}(\mathrm{UP})$ denotes the UP analog of $\mathrm{EASY}_{\forall}^{\exists}$.

Proposition 4.3.2 $\mathrm{EASY}_{\forall}^{\forall} \subseteq \mathrm{EASY}_{\forall}^{\forall}(\mathrm{FewP}) \subseteq \mathrm{EASY}_{\forall}^{\forall}(\mathrm{UP}) \subseteq \mathrm{P} \subseteq \mathrm{UP} \subseteq \mathrm{FewP} \subseteq \mathrm{NP}$.

Fenner et al. [FFNR96] have characterized the existence of surjective, many-to-one one-way functions by the condition $\mathrm{P} \not\subseteq \mathrm{EASY}_{\forall}^{\forall}$. In this section, we give analogous characterizations of the existence of surjective, one-to-one one-way functions (i.e., partial one-way permutations) and surjective, poly-to-one one-way functions by separating P from $\mathrm{EASY}_{\forall}^{\forall}(\mathrm{UP})$ and $\mathrm{EASY}_{\forall}^{\forall}(\mathrm{FewP})$, respectively.

Theorem 4.3.3 *The following four statements are equivalent.*

1. $\mathrm{EASY}_{\forall}^{\forall}(\mathrm{UP}) \neq \mathrm{P}$.
2. *There exists a partial one-way permutation.*
3. $\Sigma^* \notin \mathrm{EASY}_{\forall}^{\forall}(\mathrm{UP})$.
4. $\mathrm{EASY}_{\forall}^{\forall}(\mathrm{UP})$ *is not closed under complementation.*

Proof. (3) implies (4), since $\overline{\Sigma^*} = \emptyset$ as a finite set is in $\mathrm{EASY}_{\forall}^{\forall}(\mathrm{UP})$. (4) immediately implies (1). To see that (1) implies (3), assume there is a set $L \in \mathrm{P}$ such that $L \notin \mathrm{EASY}_{\forall}^{\forall}(\mathrm{UP})$. Let N be some UP machine accepting L such that no FP function exists that outputs the accepting path of $N(x)$ for all inputs $x \in L$. Let M be some P machine that accepts $\overline{L}$. Consider the following NPTM N': On input x, N' guesses whether $x \in L$ or $x \in \overline{L}$. If the guess was "$x \in L$," N' simulates $N(x)$; otherwise, it simulates $M(x)$. Then, N' is a

UP machine accepting Σ^*. Note that the accepting computation of $N'(x)$ for inputs $x \in L$ contains the accepting computation of $N(x)$. Since L cannot be empty (in fact, L cannot be finite, for otherwise we would have had $L \in \text{EASY}^{\forall}_{\forall}(\text{UP})$), no FP function can output, for all inputs $x \in \Sigma^*$, the accepting path of $N'(x)$. Hence, $\Sigma^* \notin \text{EASY}^{\forall}_{\forall}(\text{UP})$.

(3) implies (2): Assume $\Sigma^* \notin \text{EASY}^{\forall}_{\forall}(\text{UP})$. Let M be a UP machine accepting Σ^* such that no FP function can output the accepting path of $M(y)$ for all $y \in \Sigma^*$. For any input y, let $\text{comp}_M(y)$ denote the unique accepting path (encoded as a sequence of configurations) of $M(y)$. As in [GS88], define the function f to be

$$f(x) = \begin{cases} y & \text{if } x = \text{comp}_M(y) \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Given x, it can be checked in polynomial time whether x encodes an accepting path of M (by checking whether it starts with the initial configuration of M for some input string, all transitions from one configuration to the next are legal, and the final configuration contains an accepting final state), and if so, the input string y of M can easily be determined. Thus, $f \in$ FP. Since M is a UP machine, f is injective. The polynomial bounding the running time of M witnesses the honesty of f. Since $L(M) = \Sigma^*$, f is surjective. Finally, $f^{-1} \notin$ FP, since $f^{-1}(y) = x$ is an accepting computation of $M(y)$ for each y, and so $f^{-1} \in$ FP contradicts our assumption that M only has hard certificates. To summarize, f is a one-to-one one-way function with $\text{image}(f) = \Sigma^*$.

(2) implies (3): Let f be a one-to-one one-way function with $\text{image}(f) = \Sigma^*$. We will show that $\text{image}(f) = \Sigma^*$ is not in $\text{EASY}^{\forall}_{\forall}(\text{UP})$. Let p be the polynomial that witnesses the honesty of f. Consider the following machine M. On input y, M nondeterministically guesses all strings x of length at most $p(|y|)$, computes $f(x)$ for each guessed x, and accepts y if and only if $f(x) = y$. M is a UP machine accepting Σ^*, since f is a p-honest bijection (from some subset of Σ^* onto Σ^*) computable in polynomial time. Since $f^{-1} \notin$ FP and the accepting path of $M(y)$ contains $x = f^{-1}(y)$, no FP function can output, for all y, the accepting path of M on input y. Thus, $\Sigma^* \notin \text{EASY}^{\forall}_{\forall}(\text{UP})$. ∎

By Grollmann and Selman's [GS88] characterization of the existence of (partial) one-way permutations, we immediately have Corollary 4.3.4, which was previously proven directly by Hartmanis and Hemaspaandra (then Hemachandra) [HH88a], using different notation. Corollary 4.3.4 says that the converse of the UP analog of the Borodin-Demers theorem holds, see [HH88a] for discussion of this point. The original (i.e., NP) version of the Borodin-Demers theorem [BD76] is here stated as Theorem 3.1.1 on page 19. Moreover, note that Corollary 4.3.4 holds in every relativized world. In contrast, for the converse of the original Borodin-Demers theorem, there is a relativized counterexample, see Naor and Impagliazzo [IN88, Proposition 4.2] and also the papers [CS93, FR94, FFNR96].

As a point of interest, we note that Corollary 4.3.4 proves that separating P from a certain class containing P is equivalent to separating P from a certain class contained in P.

Corollary 4.3.4 [HH88a] $\mathrm{P} \neq \mathrm{UP} \cap \mathrm{coUP}$ *if and only if* $\mathrm{EASY}^{\forall}_{\forall}(\mathrm{UP}) \neq \mathrm{P}$.

A seemingly unrelated connection comes from finite model theory. Grädel [Grä94] has shown that $\mathrm{P} = \mathrm{UP} \cap \mathrm{coUP}$ if and only if the "weak definability principle" holds for every first order logic $\mathcal{L}$ on finite structures that captures P. The weak definability principle says: Every totally defined query (on the set of finite structures of the relations of a first order logic $\mathcal{L}$) that is implicitly definable in $\mathcal{L}$ is also explicitly definable in $\mathcal{L}$; see [Grä94] for those notions not defined here.

Corollary 4.3.5 $\mathrm{EASY}^{\forall}_{\forall}(\mathrm{UP}) \neq \mathrm{P}$ *if and only if the weak definability principle fails for some first order logic $\mathcal{L}$ on finite structures that captures P.*

Now we characterize the existence of surjective, poly-to-one one-way functions by the separation $\mathrm{P} \neq \mathrm{EASY}^{\forall}_{\forall}(\mathrm{FewP})$ and other conditions.

Theorem 4.3.6 *The following seven statements are equivalent.*

1. *There exists a surjective, poly-to-one one-way function.*
2. *There exists a total, surjective, poly-to-one one-way function.*
3. *There exists a total, poly-to-one one-way function f with* $\mathrm{image}(f) \in \mathrm{P}$.
4. *There exists a poly-to-one one-way function f with* $\mathrm{image}(f) \in \mathrm{P}$.
5. $\mathrm{EASY}^{\forall}_{\forall}(\mathrm{FewP}) \neq \mathrm{P}$.
6. $\Sigma^* \notin \mathrm{EASY}^{\forall}_{\forall}(\mathrm{FewP})$.
7. $\mathrm{EASY}^{\forall}_{\forall}(\mathrm{FewP})$ *is not closed under complementation.*

Proof. (1) implies (3), as if f is a function satisfying (1), then

$$g(x) \quad = \quad \begin{cases} 0f(x) & \text{if } f(x) \text{ is defined} \\ 1x & \text{if } f(x) \text{ is undefined} \end{cases}$$

satisfies (3). Also, (3) trivially implies (4).

(4) implies (5): Let f be a poly-to-one one-way function with $\mathrm{image}(f)$ in P. We will show that $\mathrm{image}(f)$ is not in $\mathrm{EASY}^{\forall}_{\forall}(\mathrm{FewP})$. Let p be the polynomial that witnesses the honesty of f. Consider the following machine M. On input y, M nondeterministically

guesses all strings x of length at most $p(|y|)$, computes $f(x)$ for each guessed x, and accepts y if and only if $f(x) = y$. M is a FewP machine accepting image(f), since f is a p-honest poly-to-one function computable in polynomial time. Since f is not FP-invertble and each accepting path of $M(y)$ contains some value of $f^{-1}(y)$, no FP function can output, for all y, some accepting path of M on input y. Thus, image$(f) \notin$ EASY$^{\forall}_{\forall}$(FewP).

It is clear that (2) implies (1). Suppose (1) holds, and f is a function satisfying (1). Then f' is a function satisfying (2), where

$$f'(x) \quad = \quad \begin{cases} \epsilon & \text{if } x = \epsilon \\ f(z)0 & \text{if } x = z0 \text{ and } f(z) \text{ is defined} \\ z1 & \text{if } x = z0 \text{ and } f(z) \text{ is undefined} \\ z1 & \text{if } x = z1. \end{cases}$$

The proof that conditions (5), (6), and (7) of this theorem are pairwise equivalent goes through as in the proof of the corresponding claim for EASY$^{\forall}_{\forall}$ or EASY$^{\forall}_{\forall}$(UP), see Theorem 4.3.3.

Finally, that (7) implies (1) can again be seen as in the proof of Theorem 4.3.3, the only difference being that M now is a FewP machine accepting Σ^* and the function f is now defined by $f(x) = y$ if x is *some* accepting path of $M(y)$, and $f(x)$ is undefined otherwise. Then, f is a surjective poly-to-one one-way function. This completes the proof that all statements of the theorem are equivalent. ∎

Note that P $\neq$ FewP is implied by each of the conditions of Theorem 4.3.6. Note also that P $\neq$ FewP $\cap$ coFewP implies each of the conditions of Theorem 4.3.6, though it is not known whether the converse holds. We conjecture that it does not (equivalently, we conjecture that the converse of the FewP analog of the Borodin-Demers theorem does not hold). Thus, the conditions of Theorem 4.3.6 are intermediate between the conditions P $\neq$ FewP $\cap$ coFewP and P $\neq$ FewP. Regarding the condition P $\neq$ FewP, we mention that Allender [All86] proved this condition equivalent to the existence of some very special types of poly-to-one one-way functions, and the paper [RH96] provides some more such characterizations.[5]

[5]In particular, Allender [All86] calls a poly-to-one function f *strongly* FP-*invertible* if there is a function $g \in$ FP such that for each $y \in$ image(f), $g(y)$ prints *all* elements of $f^{-1}(y)$, and he proves assertions of the form: P = FewP if and only if every total, honest FP function is strongly FP-invertible. In [RH96], a poly-to-one function f is called a *weak one-way function* if $f \in$ FP, f is honest, and f is not strongly FP-invertible. (Note: since the purpose of requiring one-way functions to be honest is to preclude the case that the FP-noninvertibility is trivial, it makes sense to require a stronger notion of honesty here—and this is the notion of honesty adopted in [All86, RH96]: f is *honest* if there exists a polynomial p such that for every $y \in$ image(f) and for *every* $x \in$ domain(f), if $y = f(x)$ then $|x| \leq p(|y|)$.) For notational convenience, in Table 4.2 below, we will say that a (poly-to-one) FP function has the "Allender property" if it satisfies this stronger definition of honesty and is not strongly FP-invertible.

Could it be the case that the conditions of Theorem 4.3.6 in fact either are equivalent to P $\neq$ FewP $\cap$ coFewP, or are equivalent to P $\neq$ FewP? Relativized counterexamples are known for each of these cases. In particular, there is a relativized world, constructed by Fortnow and Rogers [FR94], in which the conditions of Theorem 4.3.6 fail yet P $\neq$ FewP holds. Also, Lance Fortnow [For97a] has informed us that, using the techniques of Fortnow and Rogers [FR94], one can build a relativized world in which P $=$ FewP $\cap$ coFewP yet the conditions of Theorem 4.3.6 hold.

4.3.2 Total One-Way Permutations

For many types of one-way functions, the existence question has been characterized in the literature as equivalent to the separation of suitable complexity classes. Such a characterization for the existence of total one-way permutations, however, is still missing. To date, the result closest to this goal is the above-mentioned characterization of the existence of a partial, injective, and surjective one-way function f by the condition P $\neq$ UP $\cap$ coUP [GS88]. Since f is not total, f is only a bijection mapping a subset of Σ^* onto Σ^*. Thus, P $\neq$ UP $\cap$ coUP potentially is a strictly weaker condition than the existence of a total one-way permutation. Of course, such a function f can be made total [GS88], but only at the cost of loss of surjectivity (even though such a total one-way function created from f still has an image in P). However, we will show below that the existence of total one-way permutations is equivalent to the existence of total, injective one-way functions whose image is rankable (recall Definition 2.2.7 on page 17).

Theorem 4.3.7 *Total one-way permutations exist if and only if there exist total, one-to-one one-way functions whose image is rankable.*

Proof. The "only if" direction is immediate, since Σ^* is rankable.

For the converse, suppose there exists a total, one-to-one one-way function f whose image is rankable. We will define a total one-way permutation h. Intuitively, the idea is to fill in the holes in the image of f, using its rankability. Let $T = \text{image}(f)$ be rankable. For each n, let $\mathit{holes}(n) = 2^n - ||T^{=n}||$. Note that since T is rankable, *holes* is in FP. Let us introduce some useful notation. For each string x, let $k(x)$ be the lexicographical position of x among the length $|x|$ strings; e.g., $k(000) = 1$ and $k(111) = 8$. For each string x and each $j \in \mathbb{N}$, let $x - j$ denote the string that in lexicographical order comes j places before x. For each set A and each $k \in \mathbb{N}$, let $A_{[k]}$ be the kth string of A in lexicographical order. Now define the function h by

$$h(x) \quad = \quad \begin{cases} f\left(x - \sum_{i=0}^{|x|} \mathit{holes}(i)\right) & \text{if } k(x) > \mathit{holes}(|x|) \\ \left(\overline{T} \cap \Sigma^{|x|}\right)_{[k(x)]} & \text{if } k(x) \leq \mathit{holes}(|x|). \end{cases}$$

Since T is rankable and $f \in$ FP, we have $h \in$ FP. Clearly, h is honest and injective, h is total, and image$(h) = \Sigma^*$. If one could invert h in polynomial time, then f would also be FP-invertible, since the rankability of T allows one to find the string in the image of f that should be inverted with respect to h, and after inverting we shift the inverse with respect to h, say z, by $\sum_{i=0}^{|z|} \mathit{holes}(i)$ positions to obtain the true inverse with respect to f. Hence, h is a total one-way permutation. ∎

Remark 4.3.8 *Note that the rankability of the image of f suffices to give us Theorem 4.3.7, and Theorem 4.3.7 is stated in this way. However, even weaker notions would work. Without going into precise details, we remark that one just needs a function that, from some easily found and countable set of places, is an honest* address function *for the complement of the image of f, see the paper* [GHK92].

4.4 Conclusions and Open Problems

In Section 4.2, we have shown that P $\neq$ NP is a sufficient condition for strong, total, commutative AOWFs (equivalently, for strong, total, commutative AwOWFs) to exist. Since by standard techniques (namely, the natural binary-function, injectivity-not-required analog of a result of Grollmann and Selman [GS88, Sel92], see also [Ko85]), P $\neq$ NP is also a necessary condition for the existence of such functions, we obtain a complete characterization. This characterization solves the conjecture of Rabi and Sherman that strong AwOWFs exist [RS97], inasfar as one can solve it without solving the P $\stackrel{?}{=}$ NP question. Moreover, our proofs show how to construct a strong, total, commutative AOWF (equivalently, a strong, total, commutative AwOWF) from any given one-way function, which resolves an open problem of Rabi and Sherman [RS93].

We mention that most cryptographic applications are concerned with average-case complexity and randomized algorithms instead of worst-case complexity and deterministic algorithms. However, as Rabi and Sherman stress, the intriguing concept of (weakly) associative one-way functions, particularly when they are total and strong and ideally in an average-case version, may be expected to be useful in many cryptographic applications (such as in the key-agreement protocol proposed by Rivest and Sherman in 1984, see [RS97]), and may eventually offer elegant solutions to a variety of practical cryptographic problems.

We mention two open issues arising from Section 4.2. What formal claims can one prove regarding the security of the protocols of Rabi, Rivest, and Sherman? Also, in those cases where injectivity (i.e., one-to-one-ness) is known to be precluded, is poly-to-one-ness—or even two-to-one-ness—also precluded?

Partial functions	one-to-one	poly-to-one
no restriction	$P \neq UP$ [GS88]	$P \neq FewP$ [RH96]
surjective	$P \neq EASY^{\forall}_{\forall}(UP)$ (Thm. 4.3.3)	$P \neq EASY^{\forall}_{\forall}(FewP)$ (Thm. 4.3.6)
image in P	$P \neq EASY^{\forall}_{\forall}(UP)$ (Thm. 4.3.3 plus [GS88, Thm. 8])	$P \neq EASY^{\forall}_{\forall}(FewP)$ (Thm. 4.3.6)

Table 4.1: Characterizations of the existence of various types of one-way functions: the partial-function case

Total functions	one-to-one	poly-to-one
no restriction	$P \neq UP$ [GS88]	$P \neq FewP$ [All86]
surjective	open question (but note Thm. 4.3.7)	$P \neq EASY^{\forall}_{\forall}(FewP)$ (Thm. 4.3.6)
image in P	$P \neq EASY^{\forall}_{\forall}(UP)$ (Thm. 4.3.3 plus [GS88, Thm. 8])	$P \neq EASY^{\forall}_{\forall}(FewP)$ (Thm. 4.3.6)
Allender property	$P \neq UP$ [GS88]	$P \neq FewP$ [All86]
surjective & Allender property	open question (but note Thm. 4.3.7)	$P \neq FewP$ [RH96]
image in P & Allender property	$P \neq EASY^{\forall}_{\forall}(UP)$ (Thm. 4.3.3 plus [GS88, Thm. 8])	$P \neq FewP$ [RH96]

Table 4.2: Characterizations of the existence of various types of one-way functions: the total-function case

In Section 4.3, we provided a number of results that characterized the existence of partial and total one-way permutations. Tables 4.1 and 4.2 summarize the characterization results that are known from the literature and from Section 4.3.[6] Regarding Section 4.3, the ultimate goal is to find a characterization of the existence of total one-way permutations in terms of a separation of suitable complexity classes.

[6]See Footnote 5 for the explanation of what is meant by "Allender property" in Table 4.2. Note that for one-to-one functions, FP-invertibility and strong FP-invertibility (also defined in Footnote 5) are identical notions, and so the one-to-one column of Table 4.2 is not affected by the "Allender property" issue.

Chapter 5

Heuristics versus Completeness for Independent Set and Graph Coloring Problems

5.1 Introduction

The `Maximum Independent Set` problem (`MIS`, for short) is the problem of finding a maximum independent set of a given graph, i.e., a maximum subset I of vertices such that no two vertices in I are connected by an edge. The decision version of this problem, known as `Independent Set`, is one of the standard NP-complete problems, see [GJ79]. Therefore, `MIS` is considered to be an intractable problem, since there is no efficient algorithm known to solve it. However, there is a simple and efficient algorithm, the Minimum Degree Greedy Algorithm (MDG, for short), that in many cases provides satisfactory solutions of this problem. Given an input graph G, MDG chooses some vertex of minimum degree in G, adds this vertex to its output set, deletes this vertex and all its neighbors from G, and repeats this procedure until an empty graph is left.

It is known that for a number of certain well-behaved graph classes, MDG always outputs a maximum independent set of the input graph. Thus, `MIS` restricted to those graph classes is efficiently solvable. In particular, Bodlaender et al. [BTY97] note that MDG indeed outputs a maximum independent set if the input graph is a tree, a split graph, the complement of a k-tree, or a complete k-partite graph, for any k. In addition, Bodlaender et al. mention that MDG always outputs a maximum independent set when given a "well-covered" graph [BTY97].[1]

[1] A graph is said to be *well-covered* if all its maximal independent sets are of the same size, see [TT96].

What if the input graph does not have such a simple structure, and yet one employs MDG to find an independent set? Of course, the algorithm, just doing its job, is still fast and will quickly offer a solution. However, depending on the structure of the input graph, this solution might be far from optimal. How far? Is MDG at least able to *approximate* a maximum independent set of the given graph, say within a constant factor? Again, for certain graph classes (e.g., for graphs of bounded degree or of bounded average degree), it is known that MDG has a good approximation ratio [HR94].

In light of this, a reasonable venture is seeking to determine whether MDG approximates a maximum independent set of the input graph within a constant factor of r, for fixed $r \geq 1$. Since MDG chooses in each loop one vertex among the vertices of minimum degree, it makes sense to look at the worst-case and the best-case behavior of MDG.

Bodlaender, Thilikos, and Yamazaki [BTY97] define for any rational $r \geq 1$, the classes $\mathcal{A}_r$ and $\mathcal{S}_r$. $\mathcal{A}_r$ is the class of graphs for which MDG *always* (i.e., for any sequence of vertex choices) approximates a maximum independent set within a constant factor of r, and $\mathcal{S}_r$ is the class of graphs for which MDG *can* (i.e., for some sequence of vertex choices) approximate a maximum independent set within a constant factor of r.

The question investigated in [BTY97] is the following: What is the computational complexity of the recognition problems $\mathcal{A}_r$ and $\mathcal{S}_r$? We note in passing that there exist irrational numbers r for which the problems $\mathcal{S}_r$ and $\mathcal{A}_r$ are undecidable [BTY97], and thus it is reasonable to consider these problems only for rationals r.

Bodlaender et al. [BTY97] show that for each fixed rational $r \geq 1$, $\mathcal{A}_r$ is coNP-complete and $\mathcal{S}_r$ is coNP-hard. They also provide an upper bound for $\mathcal{S}_r$ by proving that, for each fixed rational $r \geq 1$, $\mathcal{S}_r$ is in P^{NP}, i.e., $\mathcal{S}_r$ can be solved by a deterministic polynomial-time Turing machine that is given access to an NP oracle. They explicitly leave open the question of whether the recognition problems $\mathcal{S}_r$, for $r > 1$, can be shown to be hard for complexity classes above NP and, optimally, whether matching upper and lower bounds for these problems can be found. For the special case of $r = 1$, they slightly improve their general coNP-hardness lower bound by showing that $\mathcal{S}_1$ is DP-hard,[2] again leaving open the question of whether this lower bound can be improved so as to match the upper bound of $\mathcal{S}_1$.

In Section 5.3, we completely settle all the questions left open in [BTY97]. Our main result (Theorem 5.3.4) pinpoints the exact computational complexity of $\mathcal{S}_r$: *For each rational* $r \geq 1$, $\mathcal{S}_r$ *is* $\mathrm{P}_{||}^{\mathrm{NP}}$*-complete*, where $\mathrm{P}_{||}^{\mathrm{NP}}$ denotes the class of sets solvable by some P machine that, instead of asking its oracle queries sequentially, accesses its NP oracle in parallel.

[2] DP [PY84] denotes the class of sets that can be represented as the difference of two NP sets. Clearly, $\mathrm{NP} \cup \mathrm{coNP} \subseteq \mathrm{DP} \subseteq \mathrm{P}^{\mathrm{NP}}$. Kadin [Kad88] has shown that the polynomial hierarchy collapses if $\mathrm{NP} = \mathrm{DP}$ or $\mathrm{coNP} = \mathrm{DP}$ or even if DP were closed under complementation.

Based on ground-breaking work by Wagner [Wag87], the class $P_{||}^{NP}$ has recently proven to be very important for determining the complexity of some extremely natural and central problems, see the survey [HHR97c]. Hemaspaandra, Hemaspaandra, and Rothe [HHR97a, HHR97b] have shown that the problem of determining the winner in Dodgson elections [Dod76]—a voting scheme proposed in 1876 by Lewis Carroll, the pen name of Charles L. Dodgson—is complete for $P_{||}^{NP}$. This result solves an open question of Bartholdi et al. [BTT89b].[3] Hemaspaandra and Wechsung [HW97] have shown that the problem `Minimum Equivalent Expression` (`MEE`, for short; see [GJ79]) is $P_{||}^{NP}$-hard, which considerably improves on the previously known coNP-hardness lower bound for this problem. It remains open whether or not $\texttt{MEE} \in P_{||}^{NP}$, and thus whether or not `MEE` is $P_{||}^{NP}$-complete. The best currently known upper bound for `MEE` is the trivial one, NP^{NP}. Variants of the `MEE` problem originally motivated introducing the polynomial hierarchy [MS72, Sto77], and determining its precise complexity, and that of its variants, is a long-standing, important open problem.

In Section 5.4, we are concerned with graph coloring. Taking a similar approach as in Section 5.3, we study the complexity of determining where (i.e., for which inputs) heuristics do well, for a number of heuristics designed for graph coloring. Graph coloring problems are of great importance in both theory and applications. The famous Four Color Conjecture was formulated in the last century, and it took generations of mathematicians to tackle this conjecture head-on until, in 1977, Appel and Haken [AH77a, AH77b] finally were able to prove it. Applications of constructing a graph coloring with as few colors as possible arise, for instance, in scheduling and partitioning problems, see [GJ79]. Unfortunately, the (optimization) problem of finding the chromatic number of a given graph is very complex, and even the (decision) problem of determining whether or not a given graph is 3-colorable (i.e., the vertices of the graph can be colored with three colors such that no two adjacent vertices have the same color) is one of the standard NP-complete problems ([Kar72], see also [Sto73, GJS76, GJ79]), thus being not efficiently solvable by current methods. However, due to there being a great deal of practical interest in finding efficient solutions—or at least good efficient approximate solutions—for these problems, it is not surprising that a large body of graph coloring heuristics have been proposed to date.

Such heuristic algorithms were analyzed in depth both from a practical and a theoretical point of view; see, e.g., the paper [MMI72] which compares certain heuristics by empirical tests on random graphs, and the work of Johnson [Joh74] which proves a number of prominent heuristics to have quite poor worst-case behavior in terms of their approximation

[3] Bartholdi et al. [BTT89b] established an NP-hardness lower bound for this problem, and asked whether it can be shown to be even NP-complete, i.e., whether it can be shown to be contained in NP. The $P_{||}^{NP}$-completeness result, however, implies that this problem cannot be NP-complete unless the polynomial hierarchy collapses down to $NP \cap coNP$.

ratio for the chromatic number. In fact, Feige and Kilian [FK96] recently proved that no deterministic polynomial-time algorithm can approximate the chromatic number within a factor of $\mathcal{O}(n^{1-\epsilon})$ for any fixed constant $\epsilon > 0$, unless NP = ZPP.[4]

Johnson's results [Joh74] are to be taken as a warning that the success or failure of a specific graph coloring heuristic strongly depends on the form of the given input graph. In Section 5.4, we study the complexity of the problem 3-Colorability when restricted to those input graphs for which a given heuristic is able to solve it.[5]

In order to formally define the problems we are interested in, fix any heuristic algorithm A for graph coloring, and define the following restriction of 3-Colorability, which we will denote A-3-Colorability: Given a graph G, can A on input G find a proper 3-coloring of G? (Again, the word "can" refers to the nondeterministic choices the algorithm has, and spells out: "does there exist some sequence of choices such that.")

In particular, we investigate the problem A-3-Colorability for a number of graph coloring heuristics A all of which are based on the sequential algorithm (sometimes called "greedy algorithm") applied to a certain vertex ordering, such as the order by decreasing degree or the recursive smallest-last order of Matula et al. [MMI72]. Other heuristics, for instance Wood's algorithm [Woo69] which we also consider, combine the sequential method with certain other strategies. We prove that the problem A-3-Colorability remains NP-complete for each heuristic A considered in Section 5.4.

This chapter is organized as follows. Section 5.2 provides some basic graph-theoretic concepts. Section 5.3 resolves the questions raised by Bodlaender et al. [BTY97] regarding independent set problems and the MDG heuristic. Section 5.4 deals with graph coloring.

5.2 Some Graph-Theoretic Concepts

All graphs considered in this chapter are undirected graphs without reflexive edges. For any graph G, let $V(G)$ denote the set of vertices of G, and let $E(G)$ denote the set of edges of G. For any vertex $v \in V(G)$, the *neighborhood* of v (denoted $N(v)$) is the set of vertices in G that are adjacent to v. For any vertex $v \in V(G)$, the *degree* of v is defined to be $||N(v)||$. For any subset $W \subseteq V(G)$, let $G[W]$ denote the subgraph of G induced by W.

Given two disjoint graphs G and H, their *union* is defined to be the graph $F = G \cup H$ with vertex set $V(F) = V(G) \cup V(H)$ and edge set $E(F) = E(G) \cup E(H)$.

[4] ZPP [Gil77] denotes the class of problems solvable in zero-error bounded probabilistic polynomial time.

[5] Usually, NP-complete graph problems are restricted with respect to certain "structural" graph properties such as planarity, bounded maximum degree, bipartiteness, etc. For instance, it is known that the problem 3-Colorability is in P when restricted to perfect graphs [GLS84], but remains NP-complete when restricted to planar graphs [Sto73]. In contrast, we restrict the problem with respect to the usefulness of a given heuristic.

For any graph G, a subset $I \subseteq V(G)$ is an *independent set* of G if for all $v, w \in I$, $\{v, w\} \notin E(G)$. An independent set is said to be a *maximum independent set* of G if it is of maximum size. For any graph G, let *mis*(G) denote the size of a maximum independent set I of G. The `Independent Set` problem (`IS`, for short) is formally defined as follows:

$$\texttt{IS} = \{\langle G, k\rangle \mid G \text{ is a graph and } k \text{ a positive integer such that } \mathit{mis}(G) \geq k\}.$$

Given a graph G, a *coloring* of G is a mapping from $V(G)$ to the positive integers, which represent the colors. A coloring ψ of G is called *proper* if for any two vertices x and y in $V(G)$, if $\{x, y\} \in E(G)$ then $\psi(x) \neq \psi(y)$. The *chromatic number* of graph G (denoted $\chi(G)$) is the minimum number of colors needed to properly color G. Given a fixed constant $k \geq 1$, graph G is said to be *k-colorable* if there exists a proper coloring of G using no more than k colors. The problem `K-Colorability` is formally defined as:

$$\texttt{K-Colorability} = \{\langle G, k\rangle \mid G \text{ is a graph and } k \text{ a positive integer such that } \chi(G) \leq k\}.$$

In Section 5.4, we will focus on the problem `3-Colorability` (the special case of `K-Colorability` with $k = 3$), which is already NP-complete.

5.3 How Hard is it to Know When Greed Can Approximate Maximum Independent Sets?

Section 5.3 is organized as follows. Section 5.3.1 formally defines the problems we are interested in. Section 5.3.2 shows that the proof of Bodlaender et al. [BTY97] to establish their upper bound of these problems in fact provides a better upper bound. Section 5.3.3 presents the reduction used to prove a matching lower bound.

5.3.1 Preliminaries

The Minimum Degree Greedy Algorithm is displayed in Figure 5.1. Let *mdg*(G) denote the maximum size of the output set of MDG on input G, where the maximum is taken over all the possible sequences of choices among the vertices of minimum degree.

For any fixed rational $r \geq 1$, $\mathcal{S}_r$ is the class of graphs for which MDG can output an independent set of size at least $1/r$ times the size of a maximum independent set. Formally,

$$\mathcal{S}_r = \{G \mid \mathit{mdg}(G) \geq \mathit{mis}(G)/r\}.$$

P^{NP} is the class of problems that can be solved by some P machine accessing an NP oracle sequentially, which means that oracle queries may depend on answers to previously

Algorithm MDG

Input A graph G
Output An independent set I of G

Set $I := \emptyset$
while $V(G) \neq \emptyset$ **do**
  Choose a vertex $v \in V(G)$ of minimum degree
  Set $I := I \cup \{v\}$
  Set $G := G[V(G) - (\{v\} \cup N(v))]$
end while

Figure 5.1: The Minimum Degree Greedy Algorithm

asked queries. $\mathrm{P}_{||}^{\mathrm{NP}}$ is the class of problems solvable by some P machine that, instead of asking its oracle queries sequentially, accesses its NP oracle in parallel. Clearly, NP $\cup$ coNP $\subseteq$ DP $\subseteq \mathrm{P}_{||}^{\mathrm{NP}} \subseteq \mathrm{P}^{\mathrm{NP}}$ and it is widely suspected that $\mathrm{P}_{||}^{\mathrm{NP}}$ differs from P^{NP}.

5.3.2 Improving the Upper Bound

Bodlaender et al. have proven that for each rational $r \geq 1$, $\mathcal{S}_r$ is in P^{NP} [BTY97, Lemma 6]. In this section we observe that for each rational $r \geq 1$, $\mathcal{S}_r$ in fact is even in $\mathrm{P}_{||}^{\mathrm{NP}}$. In fact, looking carefully at the proof of [BTY97, Lemma 6], it is easy to see that this proof already establishes the $\mathrm{P}_{||}^{\mathrm{NP}}$ upper bound. Bodlaender et al. note that for any graph G with n vertices, G does *not* belong to $\mathcal{S}_r$ if and only if there exists some k, $1 \leq k \leq n$, such that

1. $\mathit{mis}(G) \geq k$, and

2. $\mathit{mdg}(G) < k/r$, i.e., no output of MDG on input G has size at least k/r.

Given G and k, (1) can be answered by the NP-complete set `IS`, and property (2) can be checked using the NP set

$$\mathcal{T}_r = \{\langle G, k\rangle \mid \mathit{mdg}(G) \geq k/r\},$$

since (2) holds if and only if $\langle G, k\rangle \notin \mathcal{T}_r$.

Informally, the $P_{||}^{NP}$ algorithm works as follows. On input G (with $n = ||V(G)||$), the P base machine queries $\langle G, 1\rangle, \langle G, 2\rangle, \ldots, \langle G, n\rangle$ in parallel to both IS and $\mathcal{T}_r$, and accepts if and only if for some k, $\langle G, k\rangle \in$ IS and $\langle G, k\rangle \notin \mathcal{T}_r$.

Of course, we have to ensure that the P base machine queries only one NP oracle set. However, as NP is closed under disjoint union, we can take the disjoint union of IS and $\mathcal{T}_r$ and modify the queries so that they address the appropriate part of the disjoint union.

Since $P_{||}^{NP}$ is closed under complement, we have proven the following proposition.

Proposition 5.3.1 *For each rational $r \geq 1$, $\mathcal{S}_r \in P_{||}^{NP}$.*

Lowering the known upper bound of $\mathcal{S}_r$ was easy. The hard part will be raising the known lower bound of $\mathcal{S}_r$ so as to match the above upper bound.

5.3.3 Improving the Lower Bound: The Reduction

As noted in the previous section, for each rational $r \geq 1$, $\mathcal{S}_r \in P_{||}^{NP}$. Thus, to prove our main result that for each rational $r \geq 1$, $\mathcal{S}_r$ is $P_{||}^{NP}$-complete, it remains to show $P_{||}^{NP}$-hardness of $\mathcal{S}_r$. To establish this lower bound, we will reduce the problem $\mathtt{MIS}_{\mathtt{equal}}$ to $\mathcal{S}_r$, where $\mathtt{MIS}_{\mathtt{equal}}$ is defined to be the set of all pairs of graphs having maximum independent sets of the same size, i.e.,

$$\mathtt{MIS}_{\mathtt{equal}} = \{\langle G, H\rangle \mid G \text{ and } H \text{ are graphs such that } \mathit{mis}(G) = \mathit{mis}(H)\}.$$

As stated by Wagner without proof, $\mathtt{MIS}_{\mathtt{equal}}$ is $P_{||}^{NP}$-complete [Wag87].[6] A full proof of Wagner's result can be found in [HR97a].

Our reduction is given in Theorem 5.3.4 below. We will first show that $\mathtt{MIS}_{\mathtt{equal}}$ restricted to graphs in $\mathcal{S}_1$ is still $P_{||}^{NP}$-hard. This result, stated as Lemma 5.3.3 below, is the analog of Theorem 4 from [BTY97], which shows that IS restricted to graphs in $\mathcal{S}_1$ is still NP-hard. We will need the following property established in the proof of this theorem.

Lemma 5.3.2 ([BTY97], proof of Theorem 4) *Any given graph G can in polynomial time be transformed into a new graph G' such that*

1. *$G' \in \mathcal{S}_1$, and*
2. *$\mathit{mis}(G') = \mathit{mis}(G) + ||E(G)||$.*

[6] In fact, Wagner states completeness for P_{bf}^{NP}, a class that was later shown to be equivalent to $P_{||}^{NP}$, see the discussion in [KSW87, Footnote 1]. In addition, Wagner provides in [Wag90] quite a number of characterizations of $P_{||}^{NP}$, showing the robustness of this class.

The following lemma establishes that $\texttt{MIS}_{\texttt{equal}}$ restricted to graphs in $\mathcal{S}_1$ is $\mathrm{P}_{||}^{\mathrm{NP}}$-hard.

Lemma 5.3.3 *Any two graphs G and H can in polynomial time be transformed into two new graphs G' and H' such that*

1. *$G' \in \mathcal{S}_1$ and $H' \in \mathcal{S}_1$, and*

2. *mis$(G') =$ mis(H') if and only if mis$(G) =$ mis(H).*

Proof. We will first transform G and H into two new graphs G'' and H'' in such a way that G'' and H'' have the same number of edges, and such that *mis*$(G'') =$ *mis*(H'') if and only if *mis*$(G) =$ *mis*(H). The result then follows from applying Lemma 5.3.2 on G'' and H''.

Assume that $||E(G)|| \geq ||E(H)||$ and let $k = ||E(G)|| - ||E(H)||$. G'' is constructed by adding k isolated new vertices to G, and H'' is constructed by adding k isolated new edges to H. Formally,

$$\begin{aligned} V(G'') &= V(G) \cup \{x_1, \ldots, x_k\}, \\ E(G'') &= E(G), \\ V(H'') &= V(H) \cup \{y_1, \ldots, y_k, z_1, \ldots, z_k\}, \text{ and} \\ E(H'') &= E(H) \cup \{\{y_i, z_i\} \mid 1 \leq i \leq k\}, \end{aligned}$$

where the x_i's, y_i's, and z_i's are new vertices. Clearly, $||E(G'')|| = ||E(H'')||$, *mis*$(G'') =$ *mis*$(G) + k$, and *mis*$(H'') =$ *mis*$(H) + k$. Now we apply Lemma 5.3.2 on G'' and H'' to obtain two new graphs G' and H' satisfying:

1. G' and H' both are in $\mathcal{S}_1$,

2. *mis*$(G') =$ *mis*$(G'') + ||E(G'')|| =$ *mis*$(G) + k + ||E(G'')||$, and

3. *mis*$(H') =$ *mis*$(H'') + ||E(H'')|| =$ *mis*$(H) + k + ||E(H'')||$.

It is immediate that *mis*$(G') =$ *mis*(H') if and only if *mis*$(G) =$ *mis*(H). This proves the lemma. ∎

Now we state our main result.

Theorem 5.3.4 *For each rational $r \geq 1$, $\mathcal{S}_r$ is $\mathrm{P}_{||}^{\mathrm{NP}}$-complete.*

Proof. Fix any rational $r \geq 1$. Suppose $r = \frac{\ell}{m}$ for integers ℓ and m, where $\ell \geq m \geq 1$. We will define a polynomial-time computable function f that, given any pair $\langle G, H \rangle$ of graphs, outputs a graph $\widehat{G}$ such that $\langle G, H \rangle \in \texttt{MIS}_{\texttt{equal}}$ if and only if $\widehat{G} \in \mathcal{S}_r$. That is,

$$\begin{aligned} \mathit{mis}(G) = \mathit{mis}(H) &\iff mdg(\widehat{G}) \geq \mathit{mis}(\widehat{G})/r \\ &\iff \ell \cdot \mathit{mdg}(\widehat{G}) \geq m \cdot \mathit{mis}(\widehat{G}). \end{aligned}$$

First we transform G and H according to Lemma 5.3.3 into two new graphs G' and H' such that

1. G' and H' are both in $\mathcal{S}_1$, and
2. $\mathit{mis}(G') = \mathit{mis}(H')$ if and only if $\mathit{mis}(G) = \mathit{mis}(H)$.

Thus, it suffices to construct $\widehat{G}$ from G' and H' such that

$$\mathit{mis}(G') = \mathit{mis}(H') \iff \ell \cdot \mathit{mdg}(\widehat{G}) \geq m \cdot \mathit{mis}(\widehat{G}).$$

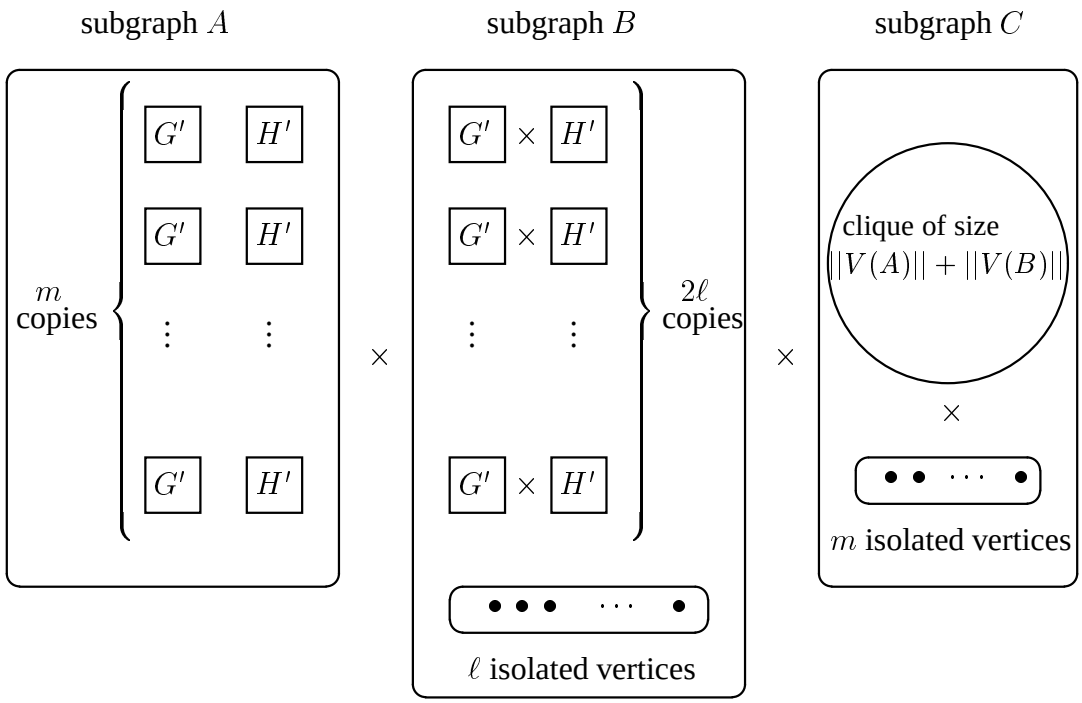

Figure 5.2: Reducing $\texttt{MIS}_{\texttt{equal}}$ to $\mathcal{S}_r$: Graph $\widehat{G}$ constructed from the graphs G' and H'

Look at Figure 5.2 for the construction of $\widehat{G}$. $\widehat{G}$ consists of $2m + 4\ell + 3$ pairwise disjoint subgraphs: $m + 2\ell$ copies of G', $m + 2\ell$ copies of H', one graph consisting of ℓ

isolated vertices, one graph consisting of m isolated vertices, and one large clique of size $||V(A)||+||V(B)||$, where A and B are subgraphs of $\widehat{G}$. All these subgraphs are connected as shown in Figure 5.2, where the symbol "$\times$" between two subgraphs denotes the union of the two subgraphs (made disjoint by renaming when necessary) with additional edges connecting each vertex of the first subgraph with each vertex of the second subgraph.[7]

More formally, let $G'_1, G'_2, \ldots, G'_{m+2\ell}, H'_1, H'_2, \ldots, H'_{m+2\ell}, B_\ell, C_m$, and C_{clique} be $2m + 4\ell + 3$ pairwise disjoint graphs such that G'_i is a copy of G', H'_i is a copy of H', B_ℓ consists of ℓ isolated vertices, C_m consist of m isolated vertices, and C_{clique} is a clique of size $(m + 2\ell)(||V(G')|| + ||V(H')||) + \ell$.

Define three graphs, A, B, and C, as follows:

$$
\begin{aligned}
V(A) &= \bigcup_{i=1}^{m} (V(G'_i) \cup V(H'_i)), \\
E(A) &= \bigcup_{i=1}^{m} (E(G'_i) \cup E(H'_i)), \\
V(B) &= V(B_\ell) \cup \bigcup_{i=m+1}^{m+2\ell} (V(G'_i) \cup V(H'_i)), \\
E(B) &= \bigcup_{i=m+1}^{m+2\ell} (E(G'_i) \cup E(H'_i) \cup \{\{x, y\} \mid x \in V(G'_i) \text{ and } y \in V(H'_i)\}), \\
V(C) &= V(C_m) \cup V(C_{clique}), \text{ and} \\
E(C) &= E(C_{clique}) \cup \{\{x, y\} \mid x \in V(C_{clique}) \text{ and } y \in V(C_m)\}.
\end{aligned}
$$

Then $\widehat{G}$ consists of the Cartesian product of its subgraphs A, B, and C, i.e.,

$$
\begin{aligned}
V(\widehat{G}) &= V(A) \cup V(B) \cup V(C), \text{ and} \\
E(\widehat{G}) &= E(A) \cup E(B) \cup E(C) \\
&\quad \cup \{\{a, b\} \mid a \in V(A) \text{ and } b \in V(B)\} \\
&\quad \cup \{\{b, c\} \mid b \in V(B) \text{ and } c \in V(C)\}.
\end{aligned}
$$

This completes the construction and defines our reduction $f(\langle G, H\rangle) = \widehat{G}$.

Clearly, f is computable in polynomial time. It remains to show that $\mathit{mis}(G') = \mathit{mis}(H')$ if and only if $\ell \cdot \mathit{mdg}(\widehat{G}) \geq m \cdot \mathit{mis}(\widehat{G})$. We will first determine $\mathit{mdg}(\widehat{G})$, the maximum size of an output set of MDG running on input $\widehat{G}$. Note that we have chosen subgraph

[7] One might be tempted to call this construction the "Cartesian product" of the two subgraphs (and the paper [HR98a] indeed used this term). However, the Cartesian product of two graphs is a standard term in graph theory and is used to denote a different notion.

C in $\widehat{G}$ large enough to enforce that MDG on input $\widehat{G}$ starts choosing vertices in subgraph A, since the maximum degree of all vertices in A is clearly less than $||V(A)|| + ||V(B)||$ and thus is smaller than the degree of any vertex in B or C. Moreover, since $G' \in \mathcal{S}_1$ and $H' \in \mathcal{S}_1$, $\mathit{mdg}(G') = \mathit{mis}(G')$ and $\mathit{mdg}(H') = \mathit{mis}(H')$. Thus, MDG on input $\widehat{G}$ picks in each copy of G' (respectively, H') contained in subgraph A exactly $\mathit{mis}(G')$ (respectively, $\mathit{mis}(H')$) vertices, deleting all the remaining vertices in A and also completely deleting subgraph B. Finally, MDG chooses all vertices in C_m and deletes the clique C_{clique}.[8]

Hence,

$$\mathit{mdg}(\widehat{G}) = m\left(\mathit{mis}(G') + \mathit{mis}(H') + 1\right).$$

Now we will determine $\mathit{mis}(\widehat{G})$, the size of a maximum independent set of $\widehat{G}$. Note that $\mathit{mis}(\widehat{G}) = \max\{\mathit{mis}(A) + \mathit{mis}(C), \mathit{mis}(B)\}$. Also, it is easy to see that

$$\begin{aligned} \mathit{mis}(A) &= m(\mathit{mis}(G') + \mathit{mis}(H')), \\ \mathit{mis}(B) &= \ell(2 \cdot \max\{\mathit{mis}(G'), \mathit{mis}(H')\} + 1), \text{ and} \\ \mathit{mis}(C) &= m. \end{aligned}$$

Since $\ell \geq m$, we have

$$\mathit{mis}(\widehat{G}) = \mathit{mis}(B) = \ell\left(2 \cdot \max\{\mathit{mis}(G'), \mathit{mis}(H')\} + 1\right).$$

It follows that

$$\begin{aligned} & \ell \cdot \mathit{mdg}(\widehat{G}) \geq m \cdot \mathit{mis}(\widehat{G}) \\ \iff\quad & \ell \cdot m\left(\mathit{mis}(G') + \mathit{mis}(H') + 1\right) \geq m \cdot \ell\left(2 \cdot \max\{\mathit{mis}(G'), \mathit{mis}(H')\} + 1\right) \\ \iff\quad & \mathit{mis}(G') + \mathit{mis}(H') + 1 \geq 2 \cdot \max\{\mathit{mis}(G'), \mathit{mis}(H')\} + 1 \\ \iff\quad & \mathit{mis}(G') + \mathit{mis}(H') \geq 2 \cdot \max\{\mathit{mis}(G'), \mathit{mis}(H')\} \\ \iff\quad & \mathit{mis}(G') = \mathit{mis}(H'), \end{aligned}$$

completing the proof. ∎

5.4 Heuristics versus Completeness for Graph Coloring

Section 5.4 is organized as follows. Section 5.4.1 presents the reduction from 3-SAT to 3-Colorability that Stockmeyer ([Sto73], see also [GJS76]) constructed to show that 3-Colorability remains NP-complete when restricted to planar graphs. This reduction will be useful in Section 5.4.2, which presents NP-completeness results for a number of restrictions of 3-Colorability to graphs for which certain graph coloring heuristics do well.

[8] If $m = 1$, C is a clique, and MDG chooses any one vertex in C.

5.4.1 The Stockmeyer Reduction from 3-Satisfiability to 3-Colorability

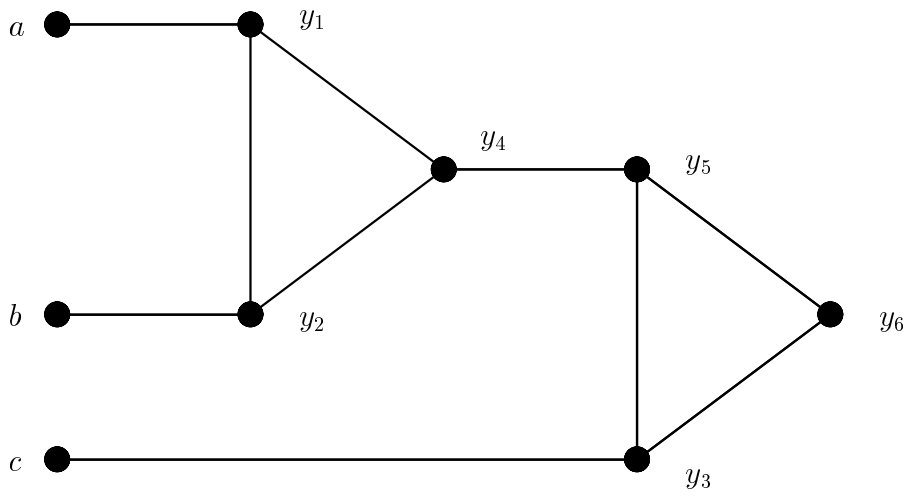

Figure 5.3: Graph H of the Stockmeyer reduction

Stockmeyer ([Sto73], see also [GJS76]) gave the following $\leq_{\mathrm{m}}^{\mathrm{p}}$-reduction from 3-SAT to 3-Colorability. We recall the Stockmeyer reduction in the present section, since it will be crucial to another reduction to be presented in Section 5.4.2.

Let ϕ be any given instance of 3-SAT, i.e., ϕ is a boolean formula in conjunctive normal form with exactly three literals per clause. Assume ϕ has n variables, $x_1, x_2, \ldots, x_n$, and m clauses, $C_1, C_2, \ldots, C_m$. The reduction maps ϕ to the graph G constructed as follows. The vertex set of G is defined to be

$$V(G) \quad = \quad \{v_1, v_2, v_3\} \cup \{x_i, \bar{x}_i \mid 1 \leq i \leq n\} \cup \{y_{jk} \mid 1 \leq j \leq m \ \wedge \ 1 \leq k \leq 6\},$$

where the x_i and $\bar{x}_i$ are vertices representing the literals x_i and $\bar{x}_i$. The edge set of G is

defined to be

$$\begin{aligned} E(G) \quad = \quad & \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_1, v_3\}\} \\ & \cup \{\{x_i, \bar{x}_i\} \mid 1 \leq i \leq n\} \\ & \cup \{\{v_3, x_i\}, \{v_3, \bar{x}_i\} \mid 1 \leq i \leq n\} \\ & \cup \{\{a_j, y_{j1}\}, \{b_j, y_{j2}\}, \{c_j, y_{j3}\} \mid 1 \leq j \leq m\} \\ & \cup \{\{v_2, y_{j6}\}, \{v_3, y_{j6}\} \mid 1 \leq j \leq m\} \\ & \cup \{\{y_{j1}, y_{j2}\}, \{y_{j1}, y_{j4}\}, \{y_{j2}, y_{j4}\} \mid 1 \leq j \leq m\} \\ & \cup \{\{y_{j3}, y_{j5}\}, \{y_{j3}, y_{j6}\}, \{y_{j5}, y_{j6}\} \mid 1 \leq j \leq m\} \\ & \cup \{\{y_{j4}, y_{j5}\} \mid 1 \leq j \leq m\}, \end{aligned}$$

where $a_j, b_j, c_j \in \bigcup_{1 \leq i \leq n} \{x_i, \bar{x}_i\}$ are vertices representing the literals occuring in clause $C_j = (a_j \vee b_j \vee c_j)$. The graph H shown in Figure 5.3 is the key construct in this reduction, which uses m disjoint copies of H (with corresponding subscripts), one for each clause C_j of ϕ. Crucially, the correctness of the reduction (i.e., ϕ is satisfiable if and only if G is 3-colorable) follows from the following properties of graph H:

(5.4.1) Any coloring of the vertices a, b, and c that assigns color 1 to one of a, b, and c can be extended to a proper 3-coloring of H that assigns color 1 to y_6.

(5.4.2) If ψ is a proper 3-coloring of H with $\psi(a) = \psi(b) = \psi(c) = i$, then $\psi(y_6) = i$.

5.4.2 The Complexity of Graph Coloring When Heuristics Do Well

Numerous heuristics for graph coloring problems have been proposed. Typically, such a heuristic consists of two parts: In the first part, a suitable ordering of the vertices of the graph is fixed; in the second part, the actual coloring algorithm is applied to the vertices in the fixed order to color the graph. A very basic coloring procedure is the so-called *sequential algorithm* (sometimes called *greedy algorithm*), which proceeds as follows. Assume the vertices of the graph are given in the order $v_1, v_2, \ldots, v_n$. Assign color 1 to v_1. For each of the remaining vertices v_i in order, assign to v_i the minimum color available, i.e., the smallest color that, so far, has not been assigned to any vertex adjacent to v_i. The sequential algorithm will be denoted by `SEQ`.

Though the local action of the sequential algorithm appears to be quite reasonable, *globally* it may fail miserably, depending on the vertex ordering chosen. Johnson [Joh74] has exhibited a sequence $G_3, \ldots, G_m, \ldots$ of graphs such that each G_m is 2-colorable, the size of G_m is linear in m, and yet the number of colors used by the sequential algorithm on input G_m is at least m for some (unfortunate) vertex ordering. Thus, for some ordering,

the sequential algorithm achieves the worst approximation ratio (of the chromatic number) possible. Johnson [Joh74] proved similar results for a number of prominent graph coloring heuristics most of which apply the sequential coloring algorithm to various vertex orderings that are obtained by seemingly reasonable procedures.

One such order-finding procedure is to order the vertices by *decreasing degree*. However, this is a rather static approach, since the place of any vertex in this ordering is independent of previously ordered vertices. A more flexible way of obtaining a vertex ordering is the recursive *smallest-last* ordering proposed by Matula et al. [MMI72], which dynamically proceeds as follows. Given a graph G with n vertices, choose any vertex of minimum degree to be the last vertex, v_n. For $i > 1$, let $v_i, \ldots, v_n$ be those vertices that have already been ordered. Choose any vertex of minimum degree in the subgraph of G induced by $V(G) - \{v_i, \ldots, v_n\}$ to be the next vertex, v_{i-1}, and proceed inductively backwards until all vertices are ordered. Note that, in both orderings, there are nondeterministic choices to be made whenever there are more vertices than one of minimum degree at any point of the procedure.

We write DD to denote (the obvious nondeterministic procedure to obtain) any ordering by decreasing degree, and we write SL to denote (the above nondeterministic procedure to obtain) any smallest-last ordering. Combining the ordering and coloring algorithms to one algorithm, A, will then specify the meta-problem A-3-Colorability defined in the introduction. For instance, combining the smallest-last ordering with the sequential algorithm, gives:

> **Decision Problem**: SL-SEQ-3-Colorability
>
> **Instance**: A graph G.
>
> **Question**: Does there exist a sequence of nondeterministic choices (between vertices of minimum degree) in the smallest-last ordering of $V(G)$ such that the sequential algorithm traversing $V(G)$ in that order properly 3-colors graph G?

First we show that the 3-Colorability problem, restricted to those input graphs on which the sequential algorithm applied to some DD vertex ordering will find a solution, is no easier to solve than the general problem.[9]

Proposition 5.4.1 DD-SEQ-3-Colorability *is* NP*-complete.*

Proof. To reduce 3-Colorability to its restriction DD-SEQ-3-Colorability, fix any graph G and a vertex of largest degree, say w, in G. W.l.o.g., assume $\mathit{deg}(w) \geq 1$. For each vertex $v \in V(G) - \{w\}$, add $\mathit{deg}(w) - \mathit{deg}(v)$ new vertices $x_{v,1}, x_{v,2}, \ldots, x_{v,\mathit{deg}(w)-\mathit{deg}(v)}$

[9] Proposition 5.4.1 clearly holds for the more general problem K-Colorability with $k \geq 3$ as well; we focus on 3-Colorability for simplicity.

to G, and connect v with each $x_{v,i}$ by an edge. Call the resulting graph G'. Then, all vertices in G' that are also vertices of G have the same degree, $deg(w)$, in G'. All new vertices $x_{v,i}$ in G' have degree 1. Assuming $G \in$ 3-Colorability, fix some proper 3-coloring, ψ, of G and define the three color classes $V_i = \{v \in V(G) \mid \psi(v) = i\}$, for $i \in \{1, 2, 3\}$, that correspond to ψ. Let $V_4 = V(G') - V(G)$ be the new vertices of G'. Since all vertices in $V_1 \cup V_2 \cup V_3$ have the same degree in G' and since all vertices in V_4 have degree 1, $V(G')$ can be DD-ordered such that the vertices of V_i come before those of V_j whenever $i < j$. This property ensures that the sequential algorithm, applied to the vertices of G' in this order, properly 3-colors G'. Conversely, if G is not 3-colorable then, by construction, G' is not 3-colorable. Hence, $G' \notin$ DD-SEQ-3-Colorability. ∎

The construction given in the proof of Proposition 5.4.1 fails for the smallest-last ordering, since the new vertices $x_{v,i}$, which are added in order to suitably increase the degree of any given vertex v relative to other vertices in G', themselves have only degree 1. Thus, they will in general occur after v in any smallest-last ordering and, as soon as they are SL-ordered, will be deleted from the graph and no longer increase the degree of v relative to other vertices still to be ordered.

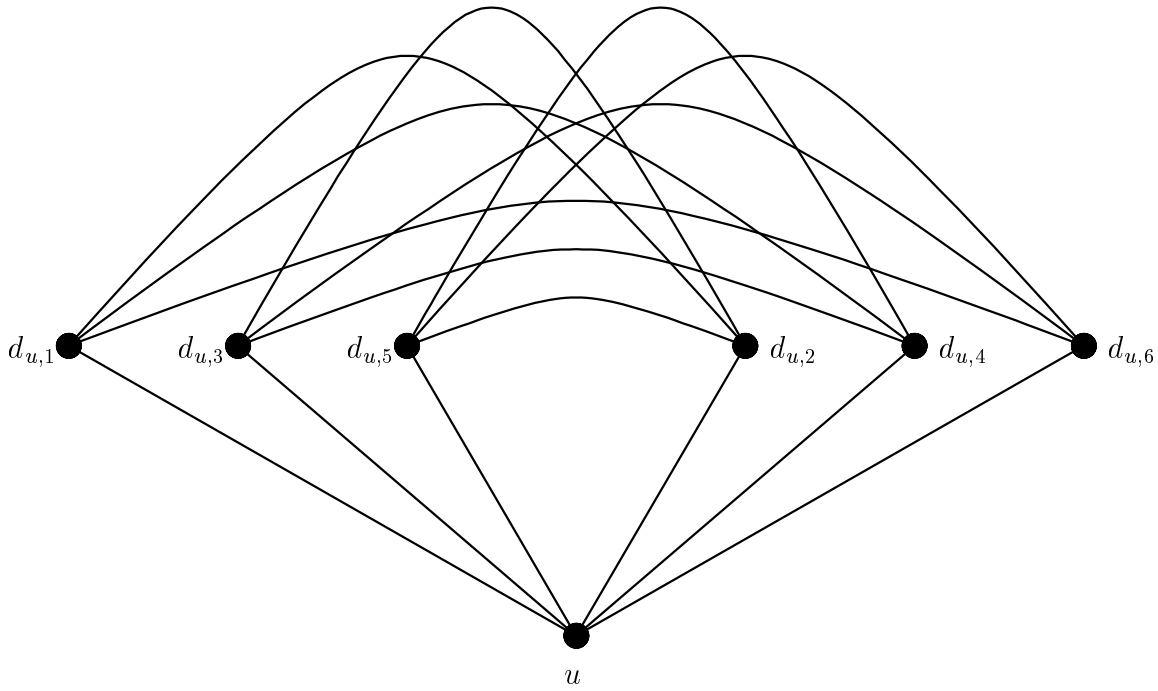

Figure 5.4: Graph $D_{u,4}$ for Lemma 5.4.2

The key construct to avoid this difficulty is given in Lemma 5.4.2 and is illustrated for a special case by graph $D_{u,4}$ shown in Figure 5.4. Consider any graph E' and suppose that some SL ordering of $V(E')$ is currently being computed, E is the subgraph of E' induced by the vertices still to be ordered, and that two vertices $u, v \in V(E)$ have a degree in E such that, say, u would be ranked above v in any SL ordering. The purpose of Lemma 5.4.2 is to show how to flip u and v in the SL ordering, assuming that the structure of E' requires such a flip for the sequential algorithm to find a proper 3-coloring of E' (if one exists).

Lemma 5.4.2 *Let E be any given graph. Let $u, v \in V(E)$ be vertices such that deg$(v) >$ deg$(u) > 0$ in E, and let $s =$ deg(v). There exists a graph $D_{u,s}$ with $V(E) \cap V(D_{u,s}) = \{u\}$ and such that*

(i) *deg$(u) >$ deg(v) in $E \cup D_{u,s}$,*

(ii) *$V(D_{u,s}) \cup \{v\}$ can be SL-ordered such that each element of $V(D_{u,s}) - \{u\}$ is ranked above v and below u, and*

(iii) *algorithm SEQ applied to this order properly 3-colors $D_{u,s}$, regardless of which color $i \in \{1, 2, 3\}$ it starts with to color u.*

Proof. Define graph $D_{u,s}$ by the vertex set

$$V(D_{u,s}) = \{u\} \cup \{d_{u,i} \mid 1 \leq i \leq 2(s-1)\}$$

and the edge set

$$E(D_{u,s}) = \{\{u, d_{u,i}\} \mid 1 \leq i \leq 2(s-1)\} \cup \{\{d_{u,i}, d_{u,j}\} \mid i \not\equiv j \bmod 2\}.$$

Note that the degree of u (relative to v) has increased in $E \cup D_{u,s}$ by $2(s-1)$. Since deg$(v) = s > 1$, this proves Property (i). Property (ii) follows from Property (i) and the fact that for each $d_{u,i}$ in $D_{u,s}$, deg$(d_{u,i}) = s$ in $E \cup D_{u,s}$: the vertex set $V(D_{u,s}) \cup \{v\}$ can be SL-ordered as $u, d_{u,1}, d_{u,2}, \ldots, d_{u,2(s-1)}, v$ in $E \cup D_{u,s}$. In particular, the sequential algorithm traversing $V(D_{u,s})$ in this order will properly 3-color $D_{u,s}$, no matter which color it starts with to color u. If color $i \in \{1, 2, 3\}$ is assigned to u, then color $1 + (i \bmod 3)$ will be assigned to all vertices $d_{u,j}$ with odd j, and color $2 + (i \bmod 3)$ will be assigned to all vertices $d_{u,j}$ with even j. This establishes Property (iii) and proves the lemma. ∎

Theorem 5.4.3 SL-SEQ-3-Colorability *is* NP*-complete.*

Proof. Instead of directly reducing 3-Colorability to SL-SEQ-3-Colorability as in Proposition 5.4.1, it is useful to base our reduction on one "generic" instance of

3-Colorability, namely on the graph G constructed by Stockmeyer ([Sto73], see also [GJS76]) to reduce 3-SAT to 3-Colorability. Simplifying the technical proof details, this approach will provide a reduction from 3-SAT to SL-SEQ-3-Colorability.

The Stockmeyer reduction, call it r, was presented in Section 5.4.1 above. Let ϕ be any given instance of 3-SAT with n variables and m clauses, and let $G = r(\phi)$ be the resulting graph. We transform G into a new graph F such that $F \in$ SL-SEQ-3-Colorability if and only if $G \in$ 3-Colorability (if and only if $\phi \in$ 3-SAT). Recall that

$$V(G) = \{v_1, v_2, v_3\} \cup \{x_i, \bar{x}_i \mid 1 \leq i \leq n\} \cup \{y_{jk} \mid 1 \leq j \leq m \ \wedge \ 1 \leq k \leq 6\}.$$

For each vertex $u \in V(G) - \{y_{j6} \mid 1 \leq j \leq m\}$, we define a graph $D_{u,s}$ associated with u as in Lemma 5.4.2, for some suitable s. Lemma 5.4.2 merely explains one local part of the overall construction; globally, the size of graph $D_{u,s}$ may affect the size of some other graph $D_{u',s'}$. The respective values of s for the various graphs $D_{u,s}$ are chosen so as to "guide" the SL algorithm so that an ordering can be obtained for which the SEQ algorithm will properly 3-color F, assuming F *is* 3-colorable.

The vertex set of graph F is given by

$$\begin{aligned} V(F) \quad = \quad & V(D_{v_1,128}) \cup V(D_{v_2,64}) \cup V(D_{v_3,128}) \\ & \cup \bigcup_{1 \leq i \leq n} V(D_{x_i,32}) \cup \bigcup_{1 \leq i \leq n} V(D_{\bar{x}_i,32}) \\ & \cup \bigcup_{1 \leq j \leq m} V(D_{y_{j1},16}) \cup \bigcup_{1 \leq j \leq m} V(D_{y_{j2},16}) \cup \bigcup_{1 \leq j \leq m} V(D_{y_{j4},8}) \\ & \cup \bigcup_{1 \leq j \leq m} V(D_{y_{j3},4}) \cup \bigcup_{1 \leq j \leq m} V(D_{y_{j5},4}) \cup \{y_{j6} \mid 1 \leq j \leq m\}. \end{aligned}$$

Note that $V(G) \subseteq V(F)$. The edge set of graph F is given by

$$\begin{aligned} E(F) \quad = \quad & E(G) \cup E(D_{v_1,128}) \cup E(D_{v_2,64}) \cup E(D_{v_3,128}) \\ & \cup \bigcup_{1 \leq i \leq n} E(D_{x_i,32}) \cup \bigcup_{1 \leq i \leq n} E(D_{\bar{x}_i,32}) \\ & \cup \bigcup_{1 \leq j \leq m} E(D_{y_{j1},16}) \cup \bigcup_{1 \leq j \leq m} E(D_{y_{j2},16}) \cup \bigcup_{1 \leq j \leq m} E(D_{y_{j4},8}) \\ & \cup \bigcup_{1 \leq j \leq m} E(D_{y_{j3},4}) \cup \bigcup_{1 \leq j \leq m} E(D_{y_{j5},4}). \end{aligned}$$

This construction yields only a linear blow-up in the size of graph F (relative to the size of G), and the reduction is polynomial-time computable.

We now argue that the construction is correct. Suppose ϕ is satisfiable (and thus G is 3-colorable). Fix some satisfying assignment $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, where $\alpha_i = 1$ if variable x_i is set to true under this assignment, and $\alpha_i = 0$ otherwise. For any literal ℓ, let $\vec{\alpha}(\ell)$ denote the value assigned to ℓ by $\vec{\alpha}$, i.e., $\vec{\alpha}(\ell) = \alpha_i$ if $\ell = x_i$, and $\vec{\alpha}(\ell) = 1 - \alpha_i$ if $\ell = \bar{x}_i$.

By construction and by Properties (i) and (ii) of Lemma 5.4.2, the vertex set of F can be SL-ordered according to Conditions (a) to (d) below. For convenience, we write $\vec{D}_{u,s}$ to denote the vertex set of graph $D_{u,s}$ given in the order $u, d_{u,1}, d_{u,2}, \ldots, d_{u,2(s-1)}$.

(a) $V(F)$ is ordered in three blocks: The first block contains the vertices

$$V(D_{v_1,128}) \cup V(D_{v_2,64}) \cup V(D_{v_3,128})$$

in the order specified by (b); the second block contains the vertices

$$\bigcup_{1 \le i \le n} (V(D_{x_i,32}) \cup V(D_{\bar{x}_i,32}))$$

in the order specified by (c); and the third block contains all the remaining vertices of F in the order specified by (d).

(b) The first block is ordered as $\vec{D}_{v_3,128}, \vec{D}_{v_1,128}, \vec{D}_{v_2,64}$.

(c) For each i, $1 \le i \le n$, the vertex set $V(D_{x_i,32}) \cup V(D_{\bar{x}_i,32})$ can be SL-ordered as $\vec{D}_{x_i,32}, \vec{D}_{\bar{x}_i,32}$ if $\alpha_i = 1$, and can be SL-ordered as $\vec{D}_{\bar{x}_i,32}, \vec{D}_{x_i,32}$ if $\alpha_i = 0$.

(d) For each j, $1 \le j \le m$, let $C_j = (a_j \vee b_j \vee c_j)$ be the jth clause of ϕ, with literals $a_j, b_j, c_j \in \bigcup_{1 \le i \le n} \{x_i, \bar{x}_i\}$. Let $\vec{\alpha}(C_j)$ be a shorthand for $(\vec{\alpha}(a_j), \vec{\alpha}(b_j), \vec{\alpha}(c_j))$. Note that since $\vec{\alpha}$ satisfies ϕ, $\vec{\alpha}(C_j) \neq (0,0,0)$ for each j. Recall that the literals in C_j are identified with the corresponding vertices of G. For each j, the vertex set associated with C_j, $Y_j = V(D_{y_{j1},16}) \cup V(D_{y_{j2},16}) \cup V(D_{y_{j3},4}) \cup V(D_{y_{j4},8}) \cup V(D_{y_{j5},4}) \cup \{y_{j6}\}$, can be SL-ordered as follows:

- **(d1)** If $\vec{\alpha}(C_j) \in \{(1,1,1), (1,1,0), (0,1,1), (0,1,0)\}$, then Y_j is ordered as $\vec{D}_{y_{j1},16}$, $\vec{D}_{y_{j2},16}$, $\vec{D}_{y_{j4},8}$, $\vec{D}_{y_{j3},4}$, $\vec{D}_{y_{j5},4}$, y_{j6}.
- **(d2)** If $\vec{\alpha}(C_j) \in \{(1,0,1), (1,0,0)\}$, then Y_j is ordered as $\vec{D}_{y_{j2},16}$, $\vec{D}_{y_{j1},16}$, $\vec{D}_{y_{j4},8}$, $\vec{D}_{y_{j3},4}$, $\vec{D}_{y_{j5},4}$, y_{j6}.
- **(d3)** If $\vec{\alpha}(C_j) \in \{(0,0,1)\}$, then Y_j is ordered as $\vec{D}_{y_{j1},16}$, $\vec{D}_{y_{j2},16}$, $\vec{D}_{y_{j4},8}$, $\vec{D}_{y_{j5},4}$, $\vec{D}_{y_{j3},4}$, y_{j6}.

The relative order between vertices not specified by Conditions (a) through (d) is irrelevant for the argument and may be fixed arbitrarily (consistent with the rules of the SL-ordering). The following property is similar to Property (5.4.1) in Section 5.4.1 and, besides, is suitably tailored to the specifics of the SEQ algorithm:

(5.4.3) For fixed j, $1 \leq j \leq m$, let the vertex set Y_j associated with clause C_j be ordered as in (d) above. Assume that the vertices representing the literals of C_j are colored such that only colors 2 and 3 are assigned and color 2 is assigned to at least one of these three vertices. Then, the SEQ algorithm properly 3-colors the subgraph of F induced by Y_j such that color 2 is assigned to y_{j6}.

This property straightforwardly follows from Property (iii) of Lemma 5.4.2 and the vertex order of Y_j given in (d).

When applied to any vertex ordering of F satisfying Conditions (a) through (d), the SEQ algorithm properly 3-colors F. In particular, it computes a coloring, ψ, of F such that $\psi(v_3) = 1$, $\psi(v_1) = 2$, $\psi(v_2) = 3$, $\psi(x_i) = 2$ and $\psi(\bar{x}_i) = 3$ if $\alpha_i = 1$, and $\psi(x_i) = 3$ and $\psi(\bar{x}_i) = 2$ if $\alpha_i = 0$, for each i with $1 \leq i \leq n$. Since $\vec{\alpha}$ is a satisfying assignment of ϕ, coloring ψ assigns color 2 to at least one (vertex representing a) literal of C_j, for each j, $1 \leq j \leq m$. By Property (5.4.3), for each j, ψ is a proper 3-coloring of the subgraph of F induced by Y_j and satisfies $\psi(y_{j6}) = 2$. Property (iii) of Lemma 5.4.2 implies that ψ (as specified so far) can be extended to a proper 3-coloring of F.

Conversely, suppose ϕ is not satisfiable (and thus G is not 3-colorable). By construction of F and by Property (5.4.2) in Section 5.4.1, this supposition implies that F is not 3-colorable. Thus, $F \notin$ SL-SEQ-3-Colorability. ∎

Both Matula et al. [MMI72] and Johnson [Joh74] have proposed generalizations of the sequential algorithm which allow the occasional *interchange* of two colors (in the coloring being computed) subject to certain sets of conditions. Johnson's algorithm, here denoted $\texttt{SEQINT}_1$, is somewhat more general than the one of Matula et al., denoted $\texttt{SEQINT}_2$, since the set of conditions under which an interchange is allowed in $\texttt{SEQINT}_2$ is slightly more restrictive than the set of conditions required in $\texttt{SEQINT}_1$ for an interchange to be performed. Both sequential-with-interchange algorithms may be combined with any vertex ordering; we focus on the DD and SL orderings. Matula et al. [MMI72] provided empirical evidence that the $\texttt{SL-SEQINT}_2$ algorithm requires significantly fewer colors than various other heuristic algorithms on random graphs.

The problems $\texttt{DD-SEQINT}_i\texttt{-3-Colorability}$ and $\texttt{SL-SEQINT}_i\texttt{-3-Colorability}$, for $i \in \{1, 2\}$, each are in NP. It should be noted that the nondeterminism here not only underlies the order-finding procedure, where more than one vertex of minimum degree may

exist, but also occurs in the coloring algorithm, where more than one bichromatic interchange may be possible. Since the $\texttt{SEQINT}_i$ algorithms include the sequential algorithm as a special case (in which no interchange is performed), we immediately have the following corollaries from respectively Proposition 5.4.1 and Theorem 5.4.3.

Corollary 5.4.4 *Both* $\texttt{DD-SEQINT}_1\texttt{-3-Colorability}$ *and* $\texttt{DD-SEQINT}_2\texttt{-3-Colorability}$ *are* NP*-complete.*

Corollary 5.4.5 *Both* $\texttt{SL-SEQINT}_1\texttt{-3-Colorability}$ *and* $\texttt{SL-SEQINT}_2\texttt{-3-Colorability}$ *are* NP*-complete.*

The last heuristic considered in this section is the algorithm of Wood [Woo69] which, given an input graph G with n vertices, proceeds in two stages as follows. In the first stage, all $n(n-1)/2$ pairs of distinct vertices are ordered by decreasing similarity, where the *similarity* of two distinct vertices x and y is defined to be

$$\textit{sim}(x,y) = \begin{cases} 0 & \text{if } \{x,y\} \in E(G) \\ ||N(x) \cap N(y)|| & \text{otherwise.} \end{cases}$$

Given this order, G will be partially colored in the first stage by executing the following steps for each pair $\{x,y\}$ in turn. In what follows, let c be a variable whose value gives the number of colors used so far.

(1) If $\textit{sim}(x,y) = 0$ then halt.

(2) If both x and y are colored, then go to next pair.

(3) If one vertex, say x, is colored, and the other one, y, is uncolored, then do the following:

- **(3a)** If $\textit{deg}(y) < c$, then go to next pair;
- **(3b)** if some vertex adjacent to y has the same color as x, then go to next pair;
- **(3c)** otherwise, assign to y the color assigned to x.

(4) If both x and y are uncolored, then do the following:

- **(4a)** If both $\textit{deg}(x) < c$ and $\textit{deg}(y) < c$, then go to next pair;
- **(4b)** otherwise, assign to both x and y the minimum color available (i.e., the smallest color $j \geq 1$ such that neither x nor y is adjacent to a vertex colored j).

After the first stage, there may remain some uncolored vertices. If so, the coloring of G will be completed in the second stage using the DD-SEQ algorithm. Wood's algorithm will be denoted by WOOD. Note that both stages of Wood's algorithm contain some amount of nondeterminism: In the first stage, we may choose between different vertex pairs of the same similarity (when there are more than one); in the second stage, we may choose among several vertices of minimum degree.

Theorem 5.4.6 WOOD-3-Colorability *is* NP-*complete.*

Proof. The proof is similar to the proof of Proposition 5.4.1, the difference being that now we have to equalize the similarity between pairs of vertices instead of the degree of vertices. Let G be any given graph. We transform G into a new graph H such that G is 3-colorable if and only if H can be 3-colored by Wood's algorithm. Let $s' = \max\{\mathit{sim}(x,y) \mid x,y \in V(G) \text{ with } x \neq y\}$ be the maximum similarity of all vertex pairs in G, and let $s = \max\{3, s'\}$.

The vertex set of H is given by

$$V(H) \quad = \quad V(G) \cup \{v' \mid v \in V(G)\} \cup \{x_{v,i} \mid v \in V(G) \ \wedge \ 1 \leq i \leq s\},$$

where the $||V(G)||$ vertices v' and the $s||V(G)||$ vertices $x_{v,i}$ are new. The edge set of H is given by

$$\begin{aligned} E(H) \quad = \quad & E(G) \cup \{\{v, x_{v,i}\} \mid v \in V(G) \ \wedge \ 1 \leq i \leq s\} \\ & \cup \{\{x_{v,i}, v'\} \mid v \in V(G) \ \wedge \ 1 \leq i \leq s\}. \end{aligned}$$

This reduction is polynomial-time computable, since the similarity of all vertex pairs in G (and hence s) can be computed in time polynomial in the size of (the encoding of) G. By construction, $\mathit{sim}(v,v') = s$ for all vertices $v \in V(G)$, and the similarity of all other vertex pairs of H is at most s. Thus, all vertex pairs of the form $\{v,v'\}$, for $v \in V(G)$, can be ranked above all other vertex pairs of H in the first stage of Wood's algorithm. Suppose G is 3-colorable. Let ψ be any fixed proper 3-coloring of G, and define the three color classes $V_i = \{v \in V(G) \mid \psi(v) = i\}$, for $i \in \{1,2,3\}$, that correspond to ψ. Let $v_1, v_2, \ldots, v_n$ be an ordering of $V(G)$ such that all vertices from V_1 come first, followed by all vertices from V_2, which in turn are followed by all vertices from V_3. Consider the corresponding order $\{v_1, v_1'\}, \{v_2, v_2'\}, \ldots, \{v_n, v_n'\}$ of the first n vertex pairs of H. Since $\mathit{deg}(v) \geq s \geq 3$ for all $v \in V(G)$ and since $\mathit{deg}(v') = s \geq 3$ for the corresponding vertices v', line (4b) of the first stage of Wood's algorithm will be executed n times and will assign color $\psi(v)$ in G to both v and v' in H. The only vertices of H as yet uncolored are those of the form $x_{v,i}$. It is then not hard to see that ψ can be extended by Wood's algorithm to a proper 3-coloring of H.

Conversely, if G is not 3-colorable, then H is not 3-colorable, and consequently $H \notin$ WOOD-3-Colorability. ∎

Finally, we mention some obvious open questions. What is the complexity of the related problem of recognizing those graphs on which a fixed heuristic can *find* the chromatic number of the graph (not just decide whether $\chi(G) \leq k$ as with K-Colorability)? What about the recognition problem for *approximating* $\chi(G)$ within a fixed factor $r \geq 1$ (r rational) of optimal? Section 5.3 did successfully resolve these questions for the Independent Set problem w.r.t. the greedy heuristic MDG, see Theorem 5.3.4 in Section 5.3. However, lower bounds for graph coloring problems in general tend to be harder to achieve than those for independent set problems. (But note that $\mathrm{P}_{||}^{\mathrm{NP}}$ also is an upper bound for these chromatic number problems—just like $\mathrm{P}_{||}^{\mathrm{NP}}$ is an upper bound for the problems $\mathcal{S}_r$.) The results of the present section may be seen as a first step towards resolving the more demanding questions raised above. In fact, the construction given in the proof of Theorem 5.3.4 is crucially based on the NP-completeness result of Bodlaender et al. [BTY97, proof of Theorem 4] (stated here as Lemma 5.3.2): The restriction of Independent Set to graphs in $\mathcal{S}_1$ is NP-complete.

Chapter 6

A Second Step Towards Complexity-Theoretic Analogs of Rice's Theorem

6.1 Introduction

The mother of complexity theory is recursive function theory. One of the most beautiful and important results of recursive function theory is Rice's Theorem. Rice's Theorem ([Ric53, Ric56], see [BS97]) states that every nontrivial language property of the recursively enumerable sets is either RE-hard or coRE-hard—and thus is certainly undecidable, a corollary that itself is often referred to as Rice's Theorem.

Theorem 6.1.1 (Rice's Theorem, Version I) *Let $\mathcal{A}$ be a nonempty proper subset of the class of recursively enumerable sets. Then either the halting problem or its complement many-one reduces to the problem: Given a Turing machine M, is $L(M) \in \mathcal{A}$?*

Corollary 6.1.2 (Rice's Theorem, Version II) *Let $\mathcal{A}$ be a nonempty proper subset of the class of recursively enumerable sets. Then the following problem is undecidable: Given a Turing machine M, is $L(M) \in \mathcal{A}$?*

Rice's Theorem conveys quite a bit of information about the nature of programs and their semantics. Programs are completely nontransparent. One can (in general) decide *nothing*—emptiness, nonemptiness, infiniteness, etc.—about the languages of given programs other than the trivial fact that each accepts some language and that language is a recursively enumerable language.[1] Recently, Kari [Kar94] has proven, for cellular automata,

[1] One must stress that Rice's Theorem refers to the languages accepted by the programs (Turing machines) rather than to machine-based actions of the programs (Turing machines)—such as whether they run for at

an analog of Rice's Theorem: All nontrivial properties of limit sets of cellular automata are undecidable.

A bold and exciting paper of Borchert and Stephan [BS97] proposes and initiates the search for *complexity-theoretic* analogs of Rice's Theorem. Borchert and Stephan note that Rice's Theorem deals with properties of programs, and they suggest as a promising complexity-theoretic analog properties of Boolean circuits. In particular, they focus on counting properties of circuits, and they point out that the parallel is a close one. Programs are concrete objects that correspond in a many-to-one way with the semantic objects, languages. Circuits (encoded into Σ^*) are concrete objects that correspond in a many-to-one way with the semantic objects, Boolean functions.

Recall that for any arity n circuit C, $\#(C)$ denotes under how many of the 2^n possible input patterns C evaluates to 1.

Definition 6.1.3 *1.* [BS97] *Each $A \subseteq \mathbb{N}$ is a* counting property of circuits. *If $A \neq \emptyset$, we say it is a nonempty property, and if $A \neq \mathbb{N}$, we say it is a proper property.*

2. [BS97] *Let A be a counting property of circuits. The* counting problem for A, $\texttt{Counting}(A)$, *is the set of all circuits C such that $\#(C) \in A$.*

3. (following usage of [BS97]) *Let A be a counting property and let $\mathcal{C}$ be a complexity class. By convention, we say that counting property A is* $\mathcal{C}$-hard *if the counting problem for A, $\texttt{Counting}(A)$, is $\mathcal{C}$-$\leq^{\mathrm{p}}_{\mathrm{T}}$-hard. (Note in particular that by this we do not mean $\mathcal{C} \subseteq \mathrm{P}^A$—we are speaking just of the complexity of A's counting problem. Note also that this convention is valid only within this chapter.)*

For succinctness and naturalness, and as it introduces no ambiguity here, throughout this chapter we use "counting" to refer to what Borchert and Stephan originally referred to as "absolute counting." For completeness, we mention that their sets $\texttt{Counting}(A)$ are not entirely new: For each A, $\texttt{Counting}(A)$ is easily seen (in light of the fact that circuits can be parsimoniously simulated by Turing machines, which themselves, as per

least seven steps on input 1776 (which is decidable) or whether for some input they do not halt (which is not decidable, but Rice's Theorem does not speak directly to this issue, that is, Rice's Theorem does not address the computability of the set $\{M \mid$ there is some input x on which $M(x)$ does not halt$\}$).

We mention in passing a related research line about "independence results in computer science." That line started with work of Hartmanis and Hopcroft [HH76] based on the nontransparency of machines, and has now reached the point where it has been shown, by Regan, that for each fixed recursively axiomatizable proof system there is a *language* with certain properties that the system cannot prove, no matter how the language is represented in the system (say, by a Turing machine accepting it). For instance, for each fixed recursively axiomatizable proof system there is a low-complexity language that is infinite, but for no Turing machine accepting the language can the proof system prove that that Turing machine accepts an infinite language. See [Reg96, Reg88] and the references therein.

the references cited in the proof of Theorem 6.3.1, can be parsimoniously transformed into Boolean formulas) to be many-one equivalent to the set, known in the literature as SAT_A or A-SAT, $\{f \mid$ the number of satisfying assignments to Boolean formula f is an integer contained in the set $A\}$ [GW87, CGH$^+$89]. Thus, $\mathtt{Counting}(A)$ inherits the various properties that the earlier papers on SAT_A established for SAT_A, such as completeness for certain counting classes. We will at times draw on this earlier work to gain insight into the properties of $\mathtt{Counting}(A)$.

The results of Borchert and Stephan that led to the research reported on in the present chapter are the following. Note that Theorem 6.1.4 is a partial analog of Theorem 6.1.1,[2] and Corollary 6.1.5 is a partial analog of Corollary 6.1.2.

Theorem 6.1.4 ([BS97], see also the comments at the start of the proof of Theorem 6.3.1) *Let A be a nonempty proper subset of $\mathbb{N}$. Then one of the following three classes is $\leq^{\mathrm{p}}_{\mathrm{m}}$-reducible to $\mathtt{Counting}(A)$:* NP, coNP, *or* UP $\oplus$ coUP.

Corollary 6.1.5 ([BS97], see also the comments at the start of the proof of Theorem 6.3.1) *Every nonempty proper counting property of circuits is* UP*-hard.*

Borchert and Stephan's paper proves a number of other results—regarding an artificial existentially quantified circuit type yielding NP-hardness, definitions and results about counting properties over rational numbers and over $\mathbb{Z}$, and so on—and we highly commend their paper to the reader. They also give a very interesting motivation. They show that, in light of the work of Valiant and Vazirani [VV86], any nontrivial counting property of circuits is hard for either NP or coNP, with respect to *randomized reductions*. Their paper and the present chapter of this thesis seek to find to what extent or in what form this behavior carries over to deterministic reductions.

The present chapter makes the following contributions. First, we extend the above-stated results of Borchert and Stephan, Theorem 6.1.4 and Corollary 6.1.5. Regarding the latter, from the same hypothesis as their Corollary 6.1.5 we derive a stronger lower bound—$\mathrm{UP}_{\mathcal{O}(1)}$-hardness. That is, we raise their lower bound from *unambiguous* nondeterminism to *low-ambiguity* nondeterminism. Second, we show that our improved lower bound cannot be further strengthened to SPP-hardness unless an unlikely complexity class containment—$\mathrm{SPP} \subseteq \mathrm{P}^{\mathrm{NP}}$—occurs. Third, we nonetheless under a very natural hypothesis raise the lower bound on the hardness of counting properties to SPP-hardness. The natural

[2] Passing on a comment from an anonymous referee of the journal paper [HR], we mention that the reader may want to also compare the UP $\oplus$ coUP occurrence in Borchert and Stephan [BS97] with the so-called Rice-Shapiro Theorem, see, e.g., [Rog67, Reg96]. We mention that in making such a comparison one should keep in mind that the Rice-Shapiro Theorem deals with showing *non-membership* in RE and coRE, rather than with showing many-one *hardness* for those classes.

hypothesis strengthens the condition on the counting property to require not merely that it is nonempty and proper, but also that it is infinite and coinfinite in a way that can be certified by polynomial-time machines.

6.2 Preliminaries

The classes UP and FewP limit the ambiguity of solutions that any instance of an NP problem potentially may have. There are various other important ambiguity-limited classes, and we will now define them, in a uniform way via counting functions. To do this, we will take the standard "#" operator—see Definition 2.2.2 on page 15—and will make it flexible enough to describe a variety of types of counting functions that are well-motivated by existing language classes. In particular, we will add a general restriction on the maximum value it can take on. For the specific case of a polynomial restriction such an operator, $\#_{\mathrm{few}}$, was already introduced by Hemaspaandra and Vollmer [HV95], see below.

Definition 6.2.1 *For each function $g : \mathbb{N} \to \mathbb{N}$ and each language class $\mathcal{C}$, define $\#_g \cdot \mathcal{C}$ to be the class of functions $f : \Sigma^* \to \mathbb{N}$ for which there exist a set $L \in \mathcal{C}$ and a polynomial s such that for each $x \in \Sigma^*$,*

$$f(x) \leq g(|x|) \text{ and } ||\{y \mid |y| = s(|x|) \text{ and } \langle x, y\rangle \in L\}|| = f(x).$$

Note that for the very special case of $\mathcal{C} = \mathrm{P}$, which is the case of importance in the present chapter, this definition simply yields classes that speak about the number of accepting paths of Turing machines that obey some constraint on their number of accepting paths. In particular, the following clearly holds for each g:

$$\#_g \cdot \mathrm{P} = \{f : \Sigma^* \to \mathbb{N} \,|\, (\exists \text{ NPTM } N)\,(\forall x \in \Sigma^*)\,[f(x) \leq g(|x|)] \text{ and } \mathrm{acc}_N(x) = f(x)\}.$$

In using Definition 6.2.1, in the case of a constant function $g(n) \equiv k$ for some $k \in \mathbb{N}$ as the ambiguity-limiting bound, we will make use of the common "λ notation" and write $\#_{\lambda n.k}$ (and will not use the simpler, though slightly informal, notation "$\#_k$").[3]

We will now define some versions of the $\#_g$ operator that focus on collections of bounds of interest to us.

Definition 6.2.2 *For each language class $\mathcal{C}$, define the following two classes of functions.*

[3]The reason we stick to this perhaps a bit cumbersome, yet formally correct, notation is to avoid notational confusion: In upcoming Chapter 8, we will provide Valiant's [Val79b] definition of the "tally" version of #P which he denoted by $\#\mathrm{P}_1$, and we will also introduce the corresponding operator which we will denote, in respect of the traditional terminology, by $\#_1$.

1. $\#_{\text{const}} \cdot \mathcal{C} = \{f : \Sigma^* \to \mathbb{N} \mid (\exists k \in \mathbb{N})\,[f \in \#_{\lambda n.k} \cdot \mathcal{C}]\}$.
2. [HV95] $\#_{\text{few}} \cdot \mathcal{C} = \{f : \Sigma^* \to \mathbb{N} \mid (\exists \text{ polynomial } s)\,[f \in \#_s \cdot \mathcal{C}]\}$.

As mentioned above, the classes UP and FewP can be written in this terminology as:

$$\begin{aligned} \text{UP} &= \{L \mid (\exists f \in \#_{\lambda n.1} \cdot \text{P})\,(\forall x \in \Sigma^*)\,[x \in L \iff f(x) > 0]\}\text{, and} \\ \text{FewP} &= \{L \mid (\exists f \in \#_{\text{few}} \cdot \text{P})\,(\forall x \in \Sigma^*)\,[x \in L \iff f(x) > 0]\} \end{aligned}$$

Now, we define a number of related ambiguity-limited classes that are well known from the literature.

Definition 6.2.3 1. ([Bei89], see also [Wat88]) *For each $k \in \mathbb{N} - \{0\}$, define*

$$\text{UP}_{\leq k} = \{L \mid (\exists f \in \#_{\lambda n.k} \cdot \text{P})\,(\forall x \in \Sigma^*)\,[x \in L \iff f(x) > 0]\}.$$

2. ([HZ93], see also [Bei89]) $\text{UP}_{\mathcal{O}(1)} = \bigcup_{k \geq 1} \text{UP}_{\leq k}$.
   (Equivalently, $\text{UP}_{\mathcal{O}(1)} = \{L \mid (\exists f \in \#_{\text{const}} \cdot \text{P})\,(\forall x \in \Sigma^*)\,[x \in L \iff f(x) > 0]\}$.*)*
3. [CH90] $\text{Few} = \text{P}^{(\#_{\text{few}} \cdot \text{P})[1]}$.
4. $\text{Const} = \text{P}^{(\#_{\text{const}} \cdot \text{P})[\mathcal{O}(1)]}$.[4]

It is well known that $\text{UP} = \text{UP}_{\leq 1} \subseteq \text{UP}_{\leq 2} \subseteq \cdots \subseteq \text{UP}_{\mathcal{O}(1)} \subseteq \text{FewP} \subseteq \text{Few} \subseteq \text{SPP}$ (the final containment is due to Köbler et al. [KSTT92], see also [FFK94] for a more general result), and clearly $\text{UP}_{\mathcal{O}(1)} \subseteq \text{Const} \subseteq \text{Few}$. Regarding relationships with the polynomial hierarchy, $\text{P} \subseteq \text{UP} \subseteq \text{FewP} \subseteq \text{NP}$, and $\text{Few} \subseteq \text{P}^{\text{FewP}}$ (so $\text{Few} \subseteq \text{P}^{\text{NP}}$). It is widely suspected that $\text{SPP} \not\subseteq \text{PH}$, though this is an open research question.

Intuitively, UP captures the notion of unambiguous nondeterminism, FewP allows polynomially ambiguous nondeterminism and, most relevant for the purposes of the present chapter, $\text{UP}_{\mathcal{O}(1)}$ allows constant-ambiguity nondeterminism. Watanabe [Wat88] has shown that $\text{P} = \text{UP}$ if and only if $\text{P} = \text{UP}_{\mathcal{O}(1)}$.

6.3 The Complexity of Counting Properties of Circuits

Now we turn to the issue of improving the known lower bounds for counting properties of circuits. Corollary 6.3.2 below raises the UP lower bound of Borchert and Stephan (Corollary 6.1.5) to a $\text{UP}_{\mathcal{O}(1)}$ lower bound. This is obtained via the even stronger bound provided by Theorem 6.3.1, which itself extends Theorem 6.1.4.

[4] As we will note in the proof of Theorem 6.3.1, $\text{P}^{(\#_{\text{const}} \cdot \text{P})[\mathcal{O}(1)]} = \text{P}^{(\#_{\text{const}} \cdot \text{P})[1]}$. Thus, the definition of Const is more analogous to the definition of Few than one might realize at first glance.

Theorem 6.3.1 *Let A be a nonempty proper subset of $\mathbb{N}$. Then one of the following three classes is $\leq^{\mathrm{p}}_{\mathrm{m}}$-reducible to* $\mathtt{Counting}(A)$: NP, coNP, *or* Const.

Corollary 6.3.2 *Every nonempty proper counting property of circuits is* $\mathrm{UP}_{\mathcal{O}(1)}$*-hard (indeed, is even* $\mathrm{UP}_{\mathcal{O}(1)}$-$\leq^{\mathrm{p}}_{1\text{-}\mathrm{tt}}$*-hard*[5]*).*

Our proof applies a constant-setting technique that Cai and Hemaspaandra [CH90] used to prove that FewP $\subseteq \oplus$P, and that Köbler et al. [KSTT92] extended to show that Few $\subseteq$ SPP. Borchert, Hemaspaandra, and Rothe [BHR99] have used the method to study the complexity of equivalence problems for OBDDs (ordered binary decision diagrams) and other structures.

Proof of Theorem 6.3.1. Let A be a nonempty proper subset of $\mathbb{N}$. The paper of Borchert and Stephan [BS97] (see Theorem 6.1.4 above) and—using different nomenclature—earlier papers [GW87, CGH$^+$89] have shown that (a) if A is finite and nonempty, then $\mathtt{Counting}(A)$ is $\leq^{\mathrm{p}}_{\mathrm{m}}$-hard for coNP, and (b) if A is cofinite and a proper subset of $\mathbb{N}$, then $\mathtt{Counting}(A)$ is $\leq^{\mathrm{p}}_{\mathrm{m}}$-hard for NP.

We will now show that if A is infinite and coinfinite, then $\mathtt{Counting}(A)$ is $\leq^{\mathrm{p}}_{\mathrm{m}}$-hard for Const. Actually, it is not hard to see that $\mathrm{P}^{(\#_{\mathrm{const}}\cdot\mathrm{P})[\mathcal{O}(1)]} = \mathrm{P}^{(\#_{\mathrm{const}}\cdot\mathrm{P})[1]}$, and so we need deal just with $\mathrm{P}^{(\#_{\mathrm{const}}\cdot\mathrm{P})[1]}$. This is a property that seems to be deeply dependent on the "const"-ness. For example, it is not known whether $\mathrm{P}^{\#\mathrm{P}[1]} = \mathrm{P}^{\#\mathrm{P}[2]}$, and indeed it is known that if this seemingly unlikely equality holds then two complexity classes associated with self-specifying machines are equal [HHW97].

The reason the equality $\mathrm{P}^{(\#_{\mathrm{const}}\cdot\mathrm{P})[\mathcal{O}(1)]} = \mathrm{P}^{(\#_{\mathrm{const}}\cdot\mathrm{P})[1]}$ holds is the following. Since each of the constant number of questions, say v, has at most a constant number of possible answers, say w, one can by brute force accept each $\mathrm{P}^{(\#_w\cdot\mathrm{P})[v]}$ language via DPTMs that make at most $u = \frac{w^{v+1}-1}{w-1}$ queries in a truth-table fashion to a function—in fact, the same function—from $\#_w \cdot \mathrm{P}$. Note that u also is a constant. Cai and Hemaspaandra [CH90] (see also [PZ83]) used a clever encoding to show that bounded-truth-table access to any #P function can be replaced by one query to a #P function. The same encoding argument shows that these u truth-table queries to the $\#_w \cdot \mathrm{P}$ function can be replaced by one query to a $\#_{w^u-1} \cdot \mathrm{P}$ function. Again, $w^u - 1$ is a constant. Hence $\mathrm{Const} = \mathrm{P}^{(\#_{\mathrm{const}}\cdot\mathrm{P})[\mathcal{O}(1)]} = \mathrm{P}^{(\#_{\mathrm{const}}\cdot\mathrm{P})[1]}$.

Let B be an arbitrary set in $\mathrm{P}^{(\#_{\mathrm{const}}\cdot\mathrm{P})[1]}$, and let $B \in \mathrm{P}^{(\#_{\mathrm{const}}\cdot\mathrm{P})[1]}$ be witnessed by some DPTM M that makes at most one query (and without loss of generality we assume that on each input x it in fact makes *exactly* one query) to some function $h \in \#_{\mathrm{const}} \cdot \mathrm{P}$. Let N' be some NPTM and let k be some constant such that for each string $z \in \Sigma^*$, $N'(z)$ has exactly $h(z)$ accepting paths and $h(z) \leq k$. Such a machine exists by the equality mentioned just

[5] Where $\leq^{\mathrm{p}}_{1\text{-}\mathrm{tt}}$ as is standard denotes polynomial-time 1-truth-table reductions [LLS75].

after Definition 6.2.1. For each input x to M, let q_x be the single query to h in the run of $M(x)$.

We will call a nonnegative integer ℓ such that $\ell \in A$ and $\ell + 1 \notin A$ a *boundary event* (of A), and we will in such cases call $\ell + 1$ a *boundary shadow*, cf. the papers [Gol89, GJY87, GHJY91]. Since A is infinite and coinfinite, note that it has infinitely many boundary events. We now define a function $g \in \#\mathrm{P}$ such that

$$(\forall x \in \Sigma^*)[M^h(x) \text{ accepts } \iff g(x) \in A]. \tag{6.3.1}$$

We will do so by mapping x for which $M^h(x)$ accepts to boundary events, and by mapping x for which $M^h(x)$ rejects to boundary shadows. To define g, we now describe an NPTM N that witnesses $g \in \#\mathrm{P}$.

On input x, N first computes the oracle query q_x of $M(x)$. Then $N(x)$ chooses $k+1$ constants $c_0, c_1, \ldots, c_k$ as follows.

$M^{\lambda z.j}(x) \in \{0,1\}$ denotes the result of the computation of $M(x)$ assuming the answer of the oracle was $h(q_x) = j$, where our convention is that $M^{\lambda z.j}(x) = 0$ stands for "reject" and $M^{\lambda z.j}(x) = 1$ stands for "accept." Let a_0 be the least boundary event of A (recall that boundary events are nonnegative integers, and thus it does make sense to speak of the least boundary event). Initially, choose

$$c_0 = \begin{cases} a_0 & \text{if } M^{\lambda z.0}(x) = 1 \\ a_0 + 1 & \text{if } M^{\lambda z.0}(x) = 0. \end{cases}$$

Successively, for $i = 1, \ldots, k$, do the following:

- Let $c_0, \ldots, c_{i-1}$ be the constants that have already been chosen. For each $i \in \mathbb{N}$, $\binom{i}{0} = 1$ as is standard. Let $b_i = \binom{i}{0}c_0 + \binom{i}{1}c_1 + \binom{i}{2}c_2 + \cdots + \binom{i}{i-1}c_{i-1}$.
- Let a_i be the least boundary event of A such that $b_i \leq a_i$.
- Set the constant

$$c_i = \begin{cases} a_i - b_i & \text{if } M^{\lambda z.i}(x) = 1 \\ a_i + 1 - b_i & \text{if } M^{\lambda z.i}(x) = 0. \end{cases}$$

After having chosen these constants,[6] $N(x)$ guesses an integer $j \in \{0, 1, ..., k\}$, and immediately splits into c_0 accepting paths if the guess was $j = 0$. For each $j > 0$ guessed,

[6]Note that as 2^{k+1} is also a constant we could alternatively simply build into the machine N a table that, for each of the 2^{k+1} behavior patterns M can have on an input (in terms of whether it accepts or rejects for each given possible answer from the oracle), states what constants $c_0, \ldots, c_k$ to use. The procedure just given would be used to decide the values of this table, which would then be hardwired into N.

$N(x)$ nondeterministically guesses each j-tuple of distinct paths of $N'(q_x)$. On each such path of $N(x)$, where the j-tuple $(\alpha_1, \alpha_2, \ldots, \alpha_j)$ of paths of $N'(q_x)$ has been guessed, $N(x)$ splits into exactly c_j accepting paths if each α_m, $1 \leq m \leq j$, is an accepting path of $N'(q_x)$. If, however, for some $1 \leq m \leq j$, α_m is a rejecting path of $N'(q_x)$, then $N(x)$ simply rejects (along the current path). This completes the description of N.

Recall that $h(q_x) \in \{0, 1, \ldots, k\}$ is the true answer of the oracle. Then, by the above construction, the number of accepting paths of $N(x)$ is

$$g(x) = c_0 + \binom{h(q_x)}{1} c_1 + \binom{h(q_x)}{2} c_2 + \cdots + \binom{h(q_x)}{h(q_x)-1} c_{h(q_x)-1} + \binom{h(q_x)}{h(q_x)} c_{h(q_x)}.$$

However, $c_{h(q_x)}$ has been chosen such that $g(x) = b_{h(q_x)} + c_{h(q_x)} = a_{h(q_x)} \in A$ if $M^h(x)$ accepts, and $g(x) = b_{h(q_x)} + c_{h(q_x)} = a_{h(q_x)} + 1 \notin A$ if $M^h(x)$ rejects. Since each a_i, $0 \leq i \leq k$, is a boundary event and each $a_i + 1$, $0 \leq i \leq k$, is a boundary shadow, this completes our proof of Equation 6.3.1.

By the well-known observation (mentioned by Garey and Johnson [GJ79, p. 169], see also the primary sources [Sim75, Val79b]) that the many-one reductions of the Cook-Karp-Levin Theorem can be altered so as to be "parsimonious," there is a $\leq^{\mathrm{p}}_{\mathrm{m}}$-reduction that on input x (N is not an input to this $\leq^{\mathrm{p}}_{\mathrm{m}}$-reduction, but rather is hardwired into the reduction) outputs a Boolean formula $\phi_x(y_1, ..., y_n)$, where n is polynomial in $|x|$, such that the number of satisfying assignments of $\phi_x(y_1, \ldots, y_n)$ equals $g(x)$. Let $C_{\phi_x}(y_1, \ldots, y_n)$ denote (the representation of) a circuit for that formula. There is a DPTM implementing this formula-to-circuit transformation. Our reduction from B to $\texttt{Counting}(A)$ is defined by $f(x) = C_{\phi_x}(y_1, \ldots, y_n)$. Clearly, f is polynomial-time computable, which together with Equation 6.3.1 implies $B \leq^{\mathrm{p}}_{\mathrm{m}} \texttt{Counting}(A)$ via f. ∎

Corollary 6.3.2 raised the lower bound of Corollary 6.1.5 from UP to $\mathrm{UP}_{\mathcal{O}(1)}$. It is natural to wonder whether the lower bound can be raised to SPP. This is especially true in light of the fact that Borchert and Stephan obtained SPP-hardness results for their notions of "counting problems over $\mathbb{Z}$" and "counting problems over the rationals"; their UP-hardness result for standard counting problems (i.e., over $\mathbb{N}$) is the short leg of their paper. However, we note that extending the hardness lower bound to SPP under the same hypothesis seems unlikely. Let BH denote the Boolean hierarchy [CGH+88]. It is well-known that NP $\subseteq$ BH $\subseteq$ P^{NP} $\subseteq$ PH.

Proposition 6.3.3 *If $A \subseteq \mathbb{N}$ is finite or cofinite, then* $\texttt{Counting}(A) \in$ BH.

This result needs no proof, as it follows easily from Lemma 3.1 and Theorem 3.1.1(a) of [CGH+89] (those results exclude the case $0 \in A$ but their proofs clearly apply also to that case) or from [GW87, Theorem 15], in light of the relationship between $\texttt{Counting}(A)$

and SAT_A mentioned earlier in the present chapter. Similarly, from earlier work one can conclude that, though for all finite and cofinite A it holds that $\mathtt{Counting}(A)$ is in the Boolean hierarchy, these problems are not good candidates for complete sets for that hierarchy's higher levels—or even its second level. In particular, from the approach of the theorem and proof of [CGH+89, Theorem 3.1.2] (see also [GW87, Theorem 15]) it is not too hard to see that $(\exists B)\,[(\forall \text{ finite } A)\,[\mathtt{Counting}(A) \text{ is not } \leq_{\mathrm{m}}^{\mathrm{p},B}\text{-hard for NP}^B] \wedge (\forall \text{ cofinite } A)\,[\mathtt{Counting}(A) \text{ is not } \leq_{\mathrm{m}}^{\mathrm{p},B}\text{-hard for coNP}^B]]$.

In light of the fact that SPP-hardness means SPP-$\leq_{\mathrm{T}}^{\mathrm{p}}$-hardness, the bound of Proposition 6.3.3 yields the following result (one can equally well state the stronger claim that no finite or cofinite counting property of circuits is SPP-$\leq_{\mathrm{m}}^{\mathrm{p}}$-hard unless SPP $\subseteq$ BH).

Corollary 6.3.4 *No finite or cofinite counting property of circuits is* SPP*-hard unless* $\mathrm{SPP} \subseteq \mathrm{P}^{\mathrm{NP}}$.

Though we do not in this thesis discuss models of relativized circuits and relativized formulas to allow this work to relativize cleanly (and we do not view this as an important issue), we mention in passing that there is a relativization in which SPP is not contained in P^{NP} (indeed, relative to which SPP strictly contains the polynomial hierarchy) [For97b].

Corollary 6.3.4 makes it clear that if we seek to prove the SPP-hardness of counting properties, we must focus only on counting properties that are simultaneously infinite and coinfinite. Even this does not seem sufficient. The problem is that there are infinite, coinfinite sets having "gaps" so huge as to make the sets have seemingly no interesting usefulness at many lengths (consider, e.g., the set $\{i \mid (\exists j)\,[i = \mathtt{AckermannFunction}(j,j)]\}$). Of course, in a recursion-theoretic context this would be no problem, as a Turing machine in the recursion-theoretic world is free from time constraints and can simply run until it finds the desired structure (which we will see is a boundary event). However, in the world of complexity theory we operate within (polynomial) time constraints. Thus, we consider it natural to add a hypothesis, in our search for an SPP-hardness result, requiring that infiniteness and coinfiniteness of a counting property be constructible in a polynomial-time manner.

Recall that a set of nonnegative integers is infinite exactly if it has no largest element. We will say that a set is P-constructibly infinite if there is a polynomial-time function that yields elements of the set at least as long as each given input.

Definition 6.3.5 1. *Let $B \subseteq \Sigma^*$. We say that B is* P-constructibly infinite *if*

$$(\exists f \in \mathrm{FP})\,(\forall x \in \Sigma^*)\,[f(x) \in B \wedge |f(x)| \geq |x|].$$

2. *Recall from Chapter 2 the standard bijection between Σ^* and $\mathbb{N}$. If $A \subseteq \mathbb{N}$, we say that A is* P*-constructibly infinite if A, viewed as a subset of Σ^* via this bijection, is* P*-constructibly infinite according to Part 1 of this definition.*

3. *If $A \subseteq \Sigma^*$ and $\overline{A}$ (or $A \subseteq \mathbb{N}$ and $\mathbb{N} - A$) are* P*-constructibly infinite, we will say that A is* P-constructibly bi-infinite.

Note that some languages that are infinite (respectively, bi-infinite) are not P-constructibly infinite (respectively, bi-infinite), e.g., languages with huge gaps between successive elements.

Borchert and Stephan [BS97] also study "counting problems over the rationals," and in this study they use a root-finding-search approach to establishing lower bounds. In the following proof, we apply this type of approach (by which we mean the successive interval contraction of the same flavor used when trying to capture the root of a function on $[a, b]$ when one knows initially that, say, $f(a) > 0$ and $f(b) < 0$) to counting problems (over $\mathbb{N}$). In particular, we use the P-constructibly bi-infinite hypothesis to "trap" a boundary event of $\overline{A}$.

Theorem 6.3.6 *Every* P*-constructibly bi-infinite counting property of circuits is* SPP*-hard.*

Proof. Let $A \subseteq \mathbb{N}$ be any P-constructibly bi-infinite counting property of circuits. Let L be any set in SPP. Since $L \in$ SPP, there are functions $f \in \#\mathrm{P}$ and $g \in$ FP such that, for each $x \in \Sigma^*$: $(x \in L \iff f(x) = 2^{|g(x)|} + 1) \wedge (x \notin L \iff f(x) = 2^{|g(x)|})$. Let h and $\overline{h}$ be FP functions certifying that A and $\overline{A}$ are P-constructibly infinite, in the exact sense of Part 2 of Definition 6.3.5. We will describe a DPTM N that $\leq^{\mathrm{p}}_{\mathrm{T}}$-reduces L to $\mathtt{Counting}(A)$. For clarity, let $\widehat{w}$ henceforth denote the natural number that in the above bijection between $\mathbb{N}$ and Σ^* corresponds to the string w. For convenience, we will sometimes view A as a subset of $\mathbb{N}$ and sometimes as a subset of Σ^* (and in the latter case we implicitly mean the transformation of A to strings under the above-mentioned bijection).

Since clearly $A \leq^{\mathrm{p}}_{\mathrm{m}} \mathtt{Counting}(A)$,[7] we for convenience will sometimes informally speak as if the set A (viewed via the bijection as a subset of Σ^*) is an oracle of the reduction. Formally, when we do so, this should be viewed as a shorthand for the complete $\leq^{\mathrm{p}}_{\mathrm{T}}$-reduction that consists of the $\leq^{\mathrm{p}}_{\mathrm{T}}$-reduction between L and A followed by the $\leq^{\mathrm{p}}_{\mathrm{m}}$-reduction between A and $\mathtt{Counting}(A)$.

We now describe the machine N. On input x, $|x| = n$, N proceeds in three steps. (As a shorthand, we will consider x fixed and will write N rather than $N^{\mathtt{Counting}(A)}(x)$.)

(1) N runs $\overline{h}$ and h on suitable inputs to find certain sufficiently large strings in $\overline{A}$ and A. In particular, let $\overline{h}(0^{|g(x)|+1}) = y$. So we have $y \notin A$ and $|y| \geq |g(x)| + 1$, and thus

[7] Either one can encode a string n (corresponding to the number $\widehat{n}$ in binary) directly into a circuit C_n such that $\#(C_n) = \widehat{n}$ (which is easy to do), or one can note the following indirect transformation: Let N' be an NPTM that on input n produces exactly $\widehat{n}$ accepting paths. Using a parsimonious Cook-Karp-Levin reduction (as described earlier), we easily obtain a family of circuits $\{\widetilde{C}_n\}_{n \in \Sigma^*}$ such that, for each $n \in \Sigma^*$, $\#(\widetilde{C}_n) = \widehat{n}$.

$\widehat{y} \geq 2^{|g(x)|+1} - 1 \geq 2^{|g(x)|}$. Recall that $|x| = n$. Since both $\overline{h}$ and g are in FP, there exists a polynomial p such that $|y| \leq p(n)$, and thus certainly $\widehat{y} < 2^{p(n)+1}$. So let $h(0^{p(n)+2}) = z$, which implies $z \in A$ and $|z| \geq p(n) + 2$. Thus, $\widehat{z} \geq 2^{p(n)+2} - 1 > 2^{p(n)+1} > \widehat{y}$. Since $h \in$ FP, there clearly exists a polynomial q such that $\widehat{z} < 2^{q(n)}$. To summarize, N has found in time polynomial in $|x|$ two strings $y \notin A$ and $z \in A$ such that $2^{|g(x)|} \leq \widehat{y} < \widehat{z} < 2^{q(n)}$.

(2) N performs a search on the interval $[\widehat{y}, \widehat{z}] \subseteq \mathbb{N}$ to find some $\widehat{u} \in \mathbb{N}$ that is a boundary event of $\overline{A}$. That is, $\widehat{u}$ will satisfy: (a) $\widehat{y} \leq \widehat{u} \leq \widehat{z}$, (b) $\widehat{u} \notin A$, and (c) $\widehat{u} + 1 \in A$. Since $\widehat{z} < 2^{q(n)}$, the search will terminate in time polynomial in $|x|$. For completeness we mention the very standard algorithm to search to find a boundary event of $\overline{A}$ (recall the comment above regarding access to A being in effect available to the algorithm):

Input $\widehat{y}$ and $\widehat{z}$ satisfying $\widehat{y} < \widehat{z}$, $\widehat{y} \notin A$, and $\widehat{z} \in A$.
Output $\widehat{u}$, a boundary event of $\overline{A}$ satisfying $\widehat{y} \leq \widehat{u} \leq \widehat{z}$.

$\widehat{u} := \widehat{y}$;
while $\widehat{z} > \widehat{u} + 1$ **do**
  $\widehat{a} := \lfloor \frac{\widehat{u}+\widehat{z}}{2} \rfloor$; **if** $\widehat{a} \notin A$ **then** $\widehat{u} := \widehat{a}$ **else** $\widehat{z} := \widehat{a}$
end while

(3) Now consider the #P function $e(\langle m, x \rangle) = m + f(x)$ and the underlying NPTM E witnessing that $e \in$ #P. Let d_E be the parsimonious Cook-Karp-Levin reduction that on each input $\langle m, x \rangle$ outputs a circuit (representation) $\widetilde{C}_{\langle m,x \rangle}$ such that $\#(\widetilde{C}_{\langle m,x \rangle}) = e(\langle m, x \rangle)$. Recall that N has already computed $\widehat{u}$ (which itself depends on x and the oracle). N, using d_E to build its query, now queries its oracle, $\texttt{Counting}(A)$, as to whether $\widetilde{C}_{\langle \widehat{u}-2^{|g(x)|},x \rangle} \in \texttt{Counting}(A)$, and N accepts its input x if and only if the answer is "yes." This completes the description of N.

As argued above, N runs in polynomial time. We have to show that it correctly $\leq^{\mathrm{p}}_{\mathrm{T}}$-reduces L to $\texttt{Counting}(A)$. Assume $x \notin L$. Then $f(x) = 2^{|g(x)|}$, and thus

$$e(\langle \widehat{u} - 2^{|g(x)|}, x \rangle) = \widehat{u} \notin A.$$

This implies that the answer to the query "$\widetilde{C}_{\langle \widehat{u}-2^{|g(x)|},x \rangle} \in \texttt{Counting}(A)$?" is "no," and so N rejects x. Analogously, if $x \in L$, then $f(x) = 2^{|g(x)|} + 1$, and thus

$$e(\langle \widehat{u} - 2^{|g(x)|}, x \rangle) = \widehat{u} + 1 \in A,$$

and so N accepts x. ∎

Finally, though we have stressed ways in which hypotheses that we feel are natural yield hardness results, we mention that for a large variety of complexity classes (amongst them SPP, BPP, PP, and FewP) one can state somewhat artificial hypotheses for A that

ensure that $\texttt{Counting}(A)$ is many-one hard for the given class. For example, if A is any set such that either $\{i \mid i \text{ is a boundary event of } A\}$ is P-constructibly infinite or $\{i \mid i \text{ is a boundary event of } \overline{A}\}$ is P-constructibly infinite, then $\texttt{Counting}(A)$ is SPP-$\leq^{\mathrm{p}}_{\mathrm{m}}$-hard.

Chapter 7

Immunity and Simplicity for Exact Counting and Other Counting Classes

7.1 Introduction

A fundamental task in complexity theory is to prove separations or collapses of complexity classes. Unfortunately, results of this kind fall short for the most important classes between polynomial time and polynomial space. In an attempt to find the reasons for this frustrating failure over many years, and to gain more insight into why these questions are beyond current techniques, researchers have studied the problem of separating complexity classes in relativized settings. Baker, Gill, and Solovay, in their seminal paper [BGS75], gave for example relativizations A and B such that $\mathrm{P}^A \neq \mathrm{NP}^A$ and $\mathrm{P}^B = \mathrm{NP}^B$, setting the stage for a host of subsequent relativization results.

Separations are also evaluated with regard to their quality. A *simple separation* such as $\mathrm{P}^A \neq \mathrm{NP}^A$ merely claims the existence of a set S in NP^A that is not recognized by any P^A machine. This can be accomplished by a simple diagonalization ensuring that every P^A machine fails to recognize S by just one string, which is put into the symmetric difference of S and the machine's language. It may well be the case, however, that some P^A machine nonetheless accepts an infinite subset of S, thus "approximating from the inside" the set witnessing the separation. Thus, one might argue that the difference between P^A and NP^A, as witnessed by S, is negligible. In contrast, a *strong separation* of P^A and NP^A is witnessed by a P^A-immune set in NP^A. Recall from Definition 2.2.5 on page 16 that, for any class $\mathcal{C}$ of sets, a set is *$\mathcal{C}$-immune* if it is an infinite set having no infinite subset in $\mathcal{C}$.

A relativization in which NP and P are strongly separated was first given by Bennett and Gill [BG81]. In fact, they prove a stronger result. Technically speaking, they show that relative to a random oracle R, NP^R contains a P^R bi-immune set with probability 1. This

was recently strengthened by Hemaspaandra and Zimand [HZ96] to the strongest result possible: Relative to a random oracle R, NP^R contains a P^R *balanced* immune set with probability 1. See these references for the notions not defined here.

Many more immunity results are known—see, e.g., the papers [HM83, SB84, Bal85, BR88, TvEB89, BJY90, Ko90, Lis, Bru92, EHTY92, BCS92]. Most important for this chapter are the results and (circuit-based) techniques of Ko [Ko90] and Bruschi [Bru92]. In particular, both papers provide relativizations in which the levels of the polynomial hierarchy (PH) separate with immunity, Bruschi's results being somewhat stronger and more refined, as they refer not only to the Σ, but also to the Δ levels of PH. Also, both authors independently obtain the result that there exists a PH-immune set in PSPACE, relative to an oracle. Since Ko's proof is only briefly sketched, Bruschi includes a detailed proof of this result. This proof, however, is flawed.[1]

Using Ko's approach, it is not difficult to give a valid and complete proof of this result (and indeed the present chapter provides such a full proof—note Corollary 7.3.6). However, the purpose of this chapter goes beyond that: We study separations with immunity for counting classes inside PSPACE with respect to the polynomial hierarchy and among each other. Counting classes that have proven particularly interesting and powerful with regard to the polynomial hierarchy are PP, $\mathrm{G{=}P}$, and $\oplus\mathrm{P}$. $\mathrm{G{=}P}$ is sometimes called the "exact counting class." Note that the PSPACE^A set that is shown by Ko [Ko90] (cf. [Bru92]) to be PH^A-immune in fact is contained in $\oplus\mathrm{P}^A$. Ko's technique [Ko90] is central to all results of the present chapter.

The relationship between these counting classes and PH still is a major open problem in complexity theory, although surprising advances have been made showing the hardness of counting. In particular, Toda [Tod91b] and Toda and Ogihara [TO92] have shown that each class $\mathcal{C}$ chosen among PP, $\mathrm{G{=}P}$, and $\oplus\mathrm{P}$ is hard for the polynomial hierarchy (and, in fact, is hard for $\mathcal{C}^{\mathrm{PH}}$) with respect to polynomial-time bounded-error random reductions. Toda [Tod91b] showed that PP is hard for PH even with respect to deterministic polynomial-time Turing reductions. However, it is widely suspected that PH is not con-

[1] In particular, looking into the proof of [Bru92, Thm. 8.3], the existence of the desired oracle extension, W, in Case (e) of the construction is not guaranteed by the circuit lower bound used. In Case (e) of Stage l, W is required to have an odd number of length $h(l)$ strings such that all circuits associated with a list of still unsatisfied requirements reject their inputs *simultaneously*—an input corresponds to the W chosen; so once W is fixed, every circuit has the same input, $\chi_W(0^{h(l)}) \cdots \chi_W(1^{h(l)})$. The used circuit lower bound for the parity function merely ensures that for each circuit C on that list, C computes parity correctly for at most 20% of the "odd" inputs of length $h(l)$. Thus, the extension W must be chosen according to the remaining 80% of such inputs to make that circuit reject. However, if there are sufficiently many circuits on the list whose correct input regions happen to cover *all* "odd" inputs of length $h(l)$ (for instance, when there are 5 circuits each being correct on a different 20% of such inputs), then there is no room left to choose a set $W \subseteq \{0,1\}^{h(l)}$ of odd cardinality that makes all circuits reject simultaneously.

tained in, and does not contain, any of these counting classes. There are oracles known relative to which each such containment fails, and similarly there are oracles relative to which each possible containment for any pair of these counting classes fails (except the known containment $\mathrm{C_{=}P} \subseteq \mathrm{PP}$ [Sim75, Wag86], which holds relative to every oracle), see [BGS75, Tor88, Tor91, Bei91, Gre91, Bei94].

Regarding relativized *strong* separations, however, the only results known are the above-mentioned result that for some A, $\oplus\mathrm{P}^A$ contains a PH^A-immune set [Ko90] (cf. [Bru92]), and that for some B, NP^B (and thus PH^B and PP^B) has a $\oplus\mathrm{P}^B$-immune set [BCS92]. In this chapter, we strengthen to relativized strong separations all the other simple separations that are possible among pairs of classes chosen from $\{\mathrm{PH}, \mathrm{PP}, \oplus\mathrm{P}, \mathrm{C_{=}P}\}$. Just as Balcázar and Russo [Bal85, BR88] exhaustively settled (in suitable relativizations) all possible immunity and simplicity questions among the probabilistic classes BPP, R, ZPP, and PP and among these classes and P and NP, we do so for the counting classes $\mathrm{C_{=}P}$, PP, and $\oplus\mathrm{P}$ among each other and with respect to the polynomial hierarchy.

Ko's proof of the result that $\oplus\mathrm{P}^A$ contains a PH^A-immune set exploits the circuit lower bounds for the parity function provided by Yao [Yao85] and Hastad [Has89]. Noticing that Hastad [Has89] proved an equally strong lower bound for the majority function, one could as well show that PP^A contains a PH^A-immune set for some oracle A. We prove a stronger result: By deriving from Razborov's [Raz87] circuit lower bound for the majority function a sufficiently strong lower bound for the Boolean function that corresponds to "exact counting," we construct an oracle relative to which even in $\mathrm{C_{=}P}$ (which is contained in PP) there exists a set that is immune even to the class $\mathrm{BPP}^{\oplus\mathrm{P}}$ (which contains PH by Toda's result [Tod91b]). This implies a number of new immunity results, including relativized $\oplus$P-immunity and PH-immunity of $\mathrm{C_{=}P}$.

Conversely, we show that, in some relativized world, NP (and thus PH and PP) contains a $\mathrm{C_{=}P}$-immune set, which strengthens Torán's simple separation of NP and $\mathrm{C_{=}P}$ [Tor88, Tor91]. As a corollary of this result, we obtain that, in the same relativization, $\mathrm{C_{=}P}$ has a *simple* set, i.e., a coinfinite $\mathrm{C_{=}P}$ set whose complement is $\mathrm{C_{=}P}$-immune. Just like immunity, the notion of simplicity originates from recursive function theory and has later proved useful also in complexity theory. The existence of a simple set in a class $\mathcal{C}$ provides strong evidence that $\mathcal{C}$ separates from the corresponding class $\mathrm{co}\mathcal{C}$. Our result that, for some oracle B, $\mathrm{C_{=}P}^B$ has a simple set extends Balcázar's result that, for some A, NP^A has a simple set [Bal85]. We also strengthen to a strong separation Green's simple separation that, relative to some oracle, $\oplus\mathrm{P} \not\subseteq \mathrm{PP}^{\mathrm{PH}}$ [Gre91]. Similarly, the relativized simple separation of the levels of the $\mathrm{PP}^{\mathrm{PH}}$ hierarchy [BU] also can be turned into a strong separation. As a special case, this includes the existence of a PP-immune set in P^{NP} (and thus in PH) relative to some oracle, which improves upon a simple separation of Beigel [Bei94].

This chapter is organized as follows. Section 7.2 recalls a useful characterization of

the polynomial hierarchy and provides some basic facts and definitions from circuit theory, see also Section 2.3. Section 7.3 presents our main result and a number of related immunity results. Section 7.4 establishes the remaining immunity results needed to provide strong relativized separations of any pair of classes chosen among PH, PP, $\oplus$P, and $\mathrm{C}_{=}\mathrm{P}$. Section 7.5 presents some open questions.

7.2 Preliminaries

As mentioned in Chapter 2, it is well-known that the levels of the polynomial hierarchy can be characterized via alternating polynomially bounded existential and universal quantifiers applied to some P predicate, and this characterization holds in the presence of any given oracle. We recall the relativized version of this characterization as the following fact to motivate the upcoming Definition 7.3.4.

Fact 7.2.1 [MS72, Sto77, Wra77] *Let A be any oracle set. For each $k \geq 0$, a set L is in $\Sigma_k^{p,A}$ if and only if there exists a polynomial p and a predicate σ computable in P^A such that for all strings x,*

$$x \in L \iff (\mathrm{Q}_1 w_1)\,(\mathrm{Q}_2 w_2)\cdots(\mathrm{Q}_k w_k)\,[\sigma(x, w_1, w_2, \ldots, w_k) = 1],$$

where the w_j range over the length $p(|x|)$ strings, and for each i, $1 \leq i \leq k$, $\mathrm{Q}_i = \exists$ if i is odd, and $\mathrm{Q}_i = \forall$ if i is even.

Some of the most important Boolean functions are the parity function and the majority function. Let us define those functions that will be considered in this work:

- $\mathrm{PAR}_n(x) = 1$ if and only if the number of bits of x that are 1 is odd.
- $\mathrm{MAJ}_n(x) = 1$ if and only if at least $\lceil \frac{n}{2} \rceil$ bits of x are 1.
- $\mathrm{EQU}_n^k(x) = 1$ if and only if exactly k bits of x are 1, where $0 \leq k \leq n$.
- $\mathrm{EQU}_n^{\mathrm{half}}(x) = 1$ if and only if exactly $\lceil \frac{n}{2} \rceil$ bits of x are 1.

Unless stated otherwise, throughout this chapter we will consider only constant depth, unbounded fanin circuits with AND, OR, and PARITY gates. Since $\{\mathrm{AND}, \mathrm{OR}, \mathrm{PARITY}\}$ (and indeed, $\{\mathrm{AND}, \mathrm{PARITY}\}$) forms a complete basis, we do not need negation gates. Note that switching from one complete basis to another increases the size of a circuit at most by a constant.

Since adjacent levels of gates of the same type can be collapsed to one level of gates of this type, we view a circuit to consist of alternating levels of respectively AND, OR, and PARITY gates, where the sequence of these operations is arbitrary—the depth of the circuit thus also measures the number of alternations.

7.3 Immunity and Simplicity Results for Exact Counting

In this section, we prove the main result of this chapter:

Theorem 7.3.1 *There exists some oracle A such that $\mathrm{G\oplus P}^A$ contains a $\mathrm{BPP}^{\oplus \mathrm{P}^A}$-immune set.*

Before turning to the actual proof, some technical details need be discussed. First, we need a sufficiently strong lower bound on the size of the "exact counting" function, $\mathrm{EQU}_n^{\mathrm{half}}$, when computed by circuits as described in the previous section. Razborov proved the following exponential lower bound on the size of the majority function when computed by such circuits; see Smolensky [Smo87] for a generalization of this result and a simplification of its proof.

Theorem 7.3.2 [Raz87] *For every k, any depth k circuit with* AND, OR, *and* PARITY *gates that computes MAJ_n has size at least $2^{\Omega(n^{1/(2k+2)})}$.*

Using this lower bound for majority, we could (by essentially the same proof as that of Theorem 7.3.1) directly establish $\mathrm{BPP}^{\oplus \mathrm{P}^A}$-immunity of PP^A. However, to obtain the stronger result of Theorem 7.3.1, we now derive from the above lower bound for majority a slightly weaker lower bound for the $\mathrm{EQU}_n^{\mathrm{half}}$ function, still being sufficiently strong to establish Theorem 7.3.1.

Lemma 7.3.3 *For every k, there exists a constant $\alpha_k > 0$ and an $n_k \in \mathbb{N}$ such that for all $n \geq n_k$, every depth k circuit with* AND, OR, *and* PARITY *gates that computes $\mathrm{EQU}_n^{\mathrm{half}}$ has size at least $n^{-1} \cdot 2^{\alpha_k n^{1/(2k+4)}}$.*

Proof. Fix a sufficiently large n. Clearly, the majority function can be expressed as $\mathrm{MAJ}_n(x) = \bigvee_{i=\lceil \frac{n}{2} \rceil}^{n} \mathrm{EQU}_n^i(x)$. Each function EQU_n^i, $0 \leq i \leq n$, is a subfunction of $\mathrm{EQU}_{2n}^{\mathrm{half}}$, since for each $x \in \{0,1\}^n$, $\mathrm{EQU}_n^i(x) = \mathrm{EQU}_{2n}^{\mathrm{half}}(x0^i1^{n-i})$. Thus, the circuit complexity of EQU_n^i is at most that of $\mathrm{EQU}_{2n}^{\mathrm{half}}$ for each i. Now let $\mathrm{size}_k(\mathrm{EQU}_n^{\mathrm{half}})$ denote the size of a smallest depth k circuit with AND, OR, and PARITY gates that computes $\mathrm{EQU}_n^{\mathrm{half}}$. By the above observation, we can realize $\mathrm{MAJ}_{\lceil \frac{n}{2} \rceil}$ with less than $n \cdot \mathrm{size}_k(\mathrm{EQU}_n^{\mathrm{half}})$ gates in depth $k+1$. Hence, by Theorem 7.3.2,

$$\mathrm{size}_k(\mathrm{EQU}_n^{\mathrm{half}}) \geq n^{-1} \cdot \mathrm{size}_{k+1}(\mathrm{MAJ}_{\lceil \frac{n}{2} \rceil}) = n^{-1} \cdot 2^{\alpha_k n^{1/(2k+4)}}$$

for some suitable constant $\alpha_k > 0$ that depends on k. ∎

For technical reasons, since we want to apply the above circuit lower bound to obtain relativized $\mathrm{BPP}^{\oplus \mathrm{P}}$-immunity, we will now give an equivalent definition of the class $\mathrm{BPP}^{\oplus \mathrm{P}}$ in terms of a hierarchy denoted $\mathrm{PH}^{\oplus}$. As explained later, $\mathrm{PH}^{\oplus}$ will only serve as a

tool in the upcoming proof of Theorem 7.3.1. $\mathrm{PH}^{\oplus}$ generalizes the polynomial hierarchy by allowing—in addition to existential and universal quantifiers—the *parity* quantifier $\bigoplus$, where $(\bigoplus w)$ means "for an odd number of strings w."

Definition 7.3.4 *Let A be any oracle set.*

1. *For each $k \geq 0$, a set L is in $\mathrm{PH}_k^{\oplus,A}$ if and only if there exists a polynomial p and a predicate σ computable in P^A such that for all strings x,*

$$x \in L \iff (\mathrm{Q}_1 w_1)\,(\mathrm{Q}_2 w_2)\,\cdots(\mathrm{Q}_k w_k)\,[\sigma(x, w_1, w_2, \ldots, w_k) = 1],$$

   where the w_j range over the length $p(|x|)$ strings and the quantifiers Q_j are chosen from $\{\exists, \forall, \bigoplus\}$.

2. *Define $\mathrm{PH}^{\oplus,A} = \bigcup_{i\geq 0} \mathrm{PH}_i^{\oplus,A}$.*

3. *We write $\mathrm{PH}_k^{\oplus}$ for $\mathrm{PH}_k^{\oplus,\emptyset}$ and $\mathrm{PH}^{\oplus}$ for $\mathrm{PH}^{\oplus,\emptyset}$.*

We stress that $\mathrm{PH}^{\oplus}$ is *not* a new complexity class or hierarchy, since it is just another name for the class $\mathrm{BPP}^{\oplus\mathrm{P}}$, as can be proven by an easy induction from the results of Toda [Tod91b] and Regan and Royer [RR95] that $\oplus\mathrm{P}^{\mathrm{BPP}^{\oplus\mathrm{P}}}$, $\mathrm{NP}^{\mathrm{BPP}^{\oplus\mathrm{P}}}$, and $\mathrm{coNP}^{\mathrm{BPP}^{\oplus\mathrm{P}}}$ each are contained in $\mathrm{BPP}^{\oplus\mathrm{P}}$.[2] Rather, the purpose of $\mathrm{PH}^{\oplus}$ is merely to simplify the proof of Theorem 7.3.1. In particular, when using $\mathrm{PH}^{\oplus}$ in place of $\mathrm{BPP}^{\oplus\mathrm{P}}$, we do not have to deal with the promise nature of BPP and, more importantly, we can straightforwardly transform circuit lower bounds for constant depth circuits over the basis $\{\mathrm{AND}, \mathrm{OR}, \mathrm{PARITY}\}$ into computations of $\mathrm{PH}_d^{\oplus}$ oracle Turing machines.

Furst, Saxe, and Sipser [FSS84] discovered the connection between computations of oracle Turing machines and circuits that allows one to transform lower bounds on the circuit complexity of Boolean functions such as parity into separations of relativized PSPACE from the relativized polynomial hierarchy. (We adopt the convention that for relativizing PSPACE, the space bound of the oracle machine be also a bound on the length of queries it may ask, for without that convention the problem of separating PSPACE^A from PH^A becomes trivial, see [FSS84].) Sufficiently strong (i.e., exponential) lower bounds for parity were then provided by Yao [Yao85] and Hastad [Has89], and were used to separate PSPACE^A from PH^A. They also proved lower bounds for variations of the Sipser functions [Sip83a] to separate all levels of PH^A from each other, see also [Ko89].

[2] In particular, due to these results, $\mathrm{PH}^{\oplus}$ in fact consists of only four levels not known to be the same: $\mathrm{PH}_0^{\oplus} = \mathrm{P}$, $\mathrm{PH}_1^{\oplus} = \mathrm{NP} \cup \mathrm{coNP} \cup \oplus\mathrm{P}, \ldots$, and $\mathrm{PH}_3^{\oplus} = \mathrm{PH}^{\oplus} = \mathrm{BPP}^{\oplus\mathrm{P}}$. Note also that in [Tod91b], Toda preferred the operator-based notation, which due to the closure of $\oplus\mathrm{P}$ under Turing reductions is equivalent, i.e., $\mathrm{BP} \cdot \oplus\mathrm{P} = \mathrm{BPP}^{\oplus\mathrm{P}}$.

A technical prerequisite for this transformation to work is that the computation of any $\Sigma_i^{p,A}$ machine can be simulated by a $\Sigma_{i+1}^{p,A}$ machine that has the property that on all computation paths at most one query is asked and this query is asked at the end of the path, see the paper [FSS84, Cor. 2.2]. An oracle machine having this property is said to be *weak*. Similarly, the computation of any $\mathrm{PH}_i^{\oplus,A}$ machine can be simulated by a weak $\mathrm{PH}_{i+1}^{\oplus,A}$ machine. The computation of a weak oracle machine M^A on some input x can then be associated with a circuit whose gates correspond to the nodes of the computation tree of $M^A(x)$, and whose inputs are the values $\chi_A(z)$ for all strings $z \in \Sigma^*$ that can be queried by $M^A(x)$. This correspondence can straightforwardly be extended to the case of weak $\mathrm{PH}^{\oplus,A}$ oracle machines and is formally stated in Proposition 7.3.5 below. The proof of Proposition 7.3.5 is standard—see, e.g., [FSS84, Lemma 2.3] and [Ko89, Lemma 2.1] for analogous results—and thus omitted. Let $\mathcal{CIR}(i,t)$ denote the collection of all depth $i+1$ circuits with AND, OR, and PARITY gates, bottom fanin at most t, and fanin at most 2^t at all remaining levels.

Proposition 7.3.5 *Let A be any oracle and let M be any weak $\mathrm{PH}_i^{\oplus,A}$ oracle machine running in time p for some polynomial p. Then, for each $x \in \Sigma^*$ of length n, there exists a circuit $C_{M,x}$ in $\mathcal{CIR}(i,p(n))$ whose inputs are the values of $\chi_A(z)$ for all strings $z \in \Sigma^*$ with $|z| \leq p(n)$ such that $C_{M,x}$ outputs 1 if and only if M^A accepts x. In particular, it follows from the bounded depth and fanin of the circuits in $\mathcal{CIR}(i,p(n))$ that the size of circuit $C_{M,x}$ is bounded by $2^{s_M(n)}$ for some polynomial s_M depending on M.*

Now we are ready to prove our main result.

Proof of Theorem 7.3.1. For any set S, let

$$L_S = \{0^N \mid N \geq 1 \text{ and the number of length } N \text{ strings in } S \text{ equals } 2^{N-1}\}.$$

Clearly, for each S, L_S is in $\mathrm{G{=}P}^S$.

We will construct the set A such that $L_A \in \mathrm{G{=}P}^A$ is $\mathrm{PH}^{\oplus,A}$-immune, i.e., L_A is infinite and no infinite subset of L_A is contained in $\mathrm{PH}^{\oplus,A}$. Since $\mathrm{BPP}^{\oplus\mathrm{P}} = \mathrm{PH}^{\oplus}$ holds true in the presence of any fixed oracle, this will prove the theorem. Also, since every $\mathrm{PH}_d^{\oplus,A}$ machine can be transformed into a weak $\mathrm{PH}_{d+1}^{\oplus,A}$ machine, it suffices to ensure in the construction of A that

(a) L_A is infinite, and

(b) for each weak $\mathrm{PH}^{\oplus,A}$ oracle machine M for which $L(M^A)$ is an infinite subset of L_A, it holds that M^A does not recognize L_A.

Fix an enumeration $M_1^{(\cdot)}, M_2^{(\cdot)}, \ldots$ of all weak $\mathrm{PH}^{\oplus,(\cdot)}$ oracle machines; we assume the machines to be clocked so that for each i, the runtime of machine $M_i^{(\cdot)}$ is bounded by

$p_i(n) = n^i + i$ for inputs of length n. In particular, if $i = \langle d, j\rangle$, the ith machine $M_i^{(\cdot)}$ in this enumeration is the jth weak $\mathrm{PH}_d^{\oplus,(\cdot)}$ oracle machine, $M_{\langle d,j\rangle}^{(\cdot)}$, in the underlying enumeration of weak $\mathrm{PH}_d^{\oplus,(\cdot)}$ oracle machines. Satisfying Property (b) above then means to satisfy in the construction the following requirement R_i for each $i \geq 1$ for which M_i^A accepts an infinite subset of L_A:

$$R_i : \quad L(M_i^A) \cap \overline{L_A} \neq \emptyset.$$

We say that Requirement R_i is *satisfied* if, at some point in the construction of A, $L(M_i^A) \cap \overline{L_A} \neq \emptyset$ can be enforced.

As a technical detail that is often used in immunity constructions, we require our enumeration of machines to satisfy that for infinitely many indices i it holds that M_i^X accepts the empty set for every oracle X, which can be assumed without loss of generality. We will need this property in order to establish (a).

Now we give the construction of A, which proceeds in stages. In Stage i, the membership in A of all strings up to length t_i will be decided, and the previous initial segment of the oracle is extended to A_i. Strings of length $\leq t_i$ that are not explicitly added to A_i are never added to the oracle. We define A to be $\bigcup_{i\geq 0} A_i$. Initially, A_0 is set to the empty set and $t_0 = 0$. Also, throughout the construction, we keep a list $\mathcal{L}$ of unsatisfied requirements. Stage $i > 0$ is as follows.

Stage i. Add i to $\mathcal{L}$. Consider all machines $M_{\ell_1}^{(\cdot)}, \ldots, M_{\ell_m}^{(\cdot)}$ corresponding to indices ℓ_r that at this point are in $\mathcal{L}$. Let $k = \max\{d_r \mid \ell_r = \langle d_r, j_r\rangle \text{ and } 1 \leq r \leq m\}$ be the maximum level of the $\mathrm{PH}^{\oplus,(\cdot)}$ hierarchy to which these machines belong (not taking into account the collapse of $\mathrm{PH}^{\oplus} = \mathrm{BPP}^{\oplus \mathrm{P}}$ mentioned in Footnote 2). Let $\alpha_{k+2} > 0$ be the constant and $n_{k+2} \in \mathbb{N}$ be the number that exist for depth $k+2$ circuits according to Lemma 7.3.3. Choose $N = N_i > \max\{t_{i-1}, \log n_{k+2}\}$ to be the smallest integer such that

$$\alpha_{k+2} \cdot 2^{N/(2k+8)} > N + i + \sum_{r=1}^{m} s_{\ell_r}(N),$$

where the polynomials $s_{\ell_r} = s_{M_{\ell_r}}$ correspond to the machines with indices in $\mathcal{L}$ according to Proposition 7.3.5.

Distinguish two cases.

Case 1: There exists an r, $1 \leq r \leq m$, and an extension $E \subseteq \Sigma^N$ of A_{i-1} such that $0^N \notin L_E$ and yet $M_{\ell_r}^{A_{i-1}\cup E}$ accepts 0^N. Let $\tilde{r}$ be the smallest such r. Cancel $\ell_{\tilde{r}}$ from $\mathcal{L}$, set A_i to $A_{i-1} \cup E$, and set t_i to $p_i(N)$. Note that Requirement $R_{\ell_{\tilde{r}}}$ has been satisfied at this stage.

Case 2: For all r, $1 \leq r \leq m$, and for all extensions $E \subseteq \Sigma^N$ of A_{i-1}, $0^N \notin L_E$ implies that $M_{\ell_r}^{A_{i-1} \cup E}$ rejects 0^N. In this case, no requirement can be satisfied at this stage. However, to achieve Property (a), we will force 0^N into L_A. Choose some extension $\tilde{E} \subseteq \Sigma^N$ of A_{i-1} such that (i) the number of length N strings in $\tilde{E}$ equals 2^{N-1}, and (ii) for each r, $1 \leq r \leq m$, $M_{\ell_r}^{A_{i-1} \cup \tilde{E}}$ rejects 0^N. We will argue later (in Claim 1 below) that such an extension $\tilde{E}$ exists. Set A_i to $A_{i-1} \cup \tilde{E}$ and set t_i to $p_i(N)$.

End of Stage i.

Note that by the definition of t_i and by our choice of N_i, the oracle extension in Stage i does not injure the computations considered in earlier stages. Thus,

$$(\forall i \geq 1) \quad [0^{N_i} \in L_{A_i} \iff 0^{N_i} \in L_A], \text{ and} \tag{7.3.1}$$

$$(\forall i, j \geq 1) \quad [M_j^{A_i} \text{ accepts } 0^{N_i} \iff M_j^A \text{ accepts } 0^{N_i}]. \tag{7.3.2}$$

The correctness of the construction will now follow from the following claims.

Claim 1. For each $i \geq 1$, there exists an oracle extension $\tilde{E}$ satisfying (i) and (ii) in Case 2 of Stage i.

Proof of Claim 1. Consider Stage i. For each $r \in \{1, \ldots, m\}$, let $C_{M_{\ell_r}, 0^N}$ be the circuit that, according to Proposition 7.3.5, corresponds to the computation of M_{ℓ_r} running on input 0^N. Fix all inputs to these circuits except those of length N consistently with A_{i-1}. That is, for each $r \in \{1, \ldots, m\}$, substitute in $C_{M_{\ell_r}, 0^N}$ the value $\chi_{A_{i-1}}(z)$ for all inputs corresponding to strings z with $|z| \leq t_{i-1}$, and substitute the value 0 for all inputs corresponding to strings z with $t_{i-1} < |z| \leq t_i$ and $|z| \neq N$. Call the resulting circuits $\widehat{C}_{\ell_1, 0^N}, \ldots, \widehat{C}_{\ell_m, 0^N}$. By Proposition 7.3.5, for each r, $\widehat{C}_{\ell_r, 0^N}$ is in $\mathcal{CIR}(k, p_{\ell_r}(N))$, its 2^N inputs correspond to the length N strings, and for each $E \subseteq \Sigma^N$, it holds that

$$\widehat{C}_{\ell_r, 0^N} \text{ on input } \chi_E(0^N) \cdots \chi_E(1^N) \text{ outputs 1} \iff M_{\ell_r}^{A_{i-1} \cup E} \text{ accepts } 0^N. \tag{7.3.3}$$

Create a new circuit $C_{2^N} = \mathrm{OR}_{r=1}^m \widehat{C}_{M_{\ell_r}, 0^N}$ whose 2^N inputs correspond to the length N strings and whose output gate is an OR gate over the subcircuits $\widehat{C}_{\ell_1, 0^N}, \ldots, \widehat{C}_{\ell_m, 0^N}$. Thus, C_{2^N} is a depth $k+2$ circuit with AND, OR, and $\oplus$ gates whose size is bounded by

$$1 + \sum_{r=1}^{m} 2^{s_{\ell_r}(N)} \leq 2^{i + \sum_{r=1}^m s_{\ell_r}(N)}$$

(note that $m \leq i$). By our choice of N, we have $2^N > n_{k+2}$ and

$$2^{i + \sum_{r=1}^m s_{\ell_r}(N)} < 2^{-N} \cdot 2^{\alpha_{k+2}(2^N)^{1/(2k+8)}}.$$

Thus, by Lemma 7.3.3, circuit C_{2^N} cannot compute the function $\mathrm{EQU}_{2^N}^{\mathrm{half}}$ correctly for all inputs. Since by the condition stated in Case 2 and by Equivalence (7.3.3) above, C_{2^N} behaves correctly for all inputs corresponding to any set E of length N strings with $0^N \notin L_E$, it follows that C_{2^N} must be incorrect on an input corresponding to some set $\tilde{E}$ of length N strings with $0^N \in L_{\tilde{E}}$, i.e., C_{2^N} on input $\chi_{\tilde{E}}(0^N) \cdots \chi_{\tilde{E}}(1^N)$ outputs 0. Since C_{2^N} is the OR of its subcircuits, each subcircuit outputs 0 on this input. Thus, Equivalence (7.3.3) implies that for each r, $1 \leq r \leq m$, $M_{\ell_r}^{A_{i-1} \cup \tilde{E}}$ rejects 0^N. ∎ Claim 1

Claim 2. L_A is an infinite set.

Proof of Claim 2. Recall our assumption that the index set of the empty set is infinite. Since no requirement R_i for which i is an index of the empty set can ever be satisfied and since, by construction, some requirement is satisfied whenever Case 1 occurs, this assumption implies that Case 2 must happen infinitely often. By construction, some string is forced into L_A whenever Case 2 occurs. Hence, L_A is an infinite set. This proves the claim and establishes Property (a). ∎ Claim 2

Claim 3. For every $i \geq 1$, M_i^A does not accept an infinite subset of L_A.

Proof of Claim 3. For each i, Requirement R_i either is satisfied at some stage of the construction, or is never satisfied. If R_i is satisfied at Stage j, then Case 1 happens in Stage j, and so $0^{N_j} \in L(M_i^{A_j}) \cap \overline{L_{A_j}}$. By Equivalences (7.3.1) and (7.3.2), $0^{N_j} \in L(M_i^A) \cap \overline{L_A}$, so $L(M_i^A) \not\subseteq L_A$. Now suppose that Requirement R_i is never satisfied. We will argue that $L(M_i^A) \cap L_A$ then is a finite set. By construction, since we added to A only strings of lengths N_j, where $j \geq 1$ and N_j is the integer chosen in Stage j, L_A contains only strings of the form 0^{N_j} for some $j \geq 1$. Note that i is added to $\mathcal{L}$ in Stage i and will stay there forever. For each $j \geq i$, if $0^{N_j} \in L_A$ (and thus $0^{N_j} \in L_{A_j}$ by (7.3.1)), then Case 2 must have occurred in Stage j. Consequently, $M_i^{A_j}$ (and thus M_i^A by (7.3.2)) rejects 0^{N_j} for every $j \geq i$. It follows that for each i, $L(M_i^A) \cap L_A$ has at most $i-1$ elements, proving the claim. ∎ Claim 3

Hence, L_A is a $\mathrm{BPP}^{\oplus \mathrm{P}^A}$-immune set in $\mathrm{C_{=}P}^A$. ∎

In particular, Theorem 7.3.1 immediately gives the following corollary. All strong separations in Corollary 7.3.6 are new, except the PH^A-immunity of PSPACE^A (and of $\mathrm{P}^{\mathrm{PP}^A}$, since $(\forall B)\,[\oplus \mathrm{P}^B \subseteq \mathrm{P}^{\mathrm{PP}^B}]$), which is also stated (or is implicit) in [Ko90, Bru92], and except the BPP^C-immunity of PP^C (and its superclasses) proven in [BR88]. We also mention that Bovet et al. [BCS92] noted that PP^D strongly separates from $\Sigma_2^{p,D}$ for some oracle D.

Corollary 7.3.6 *Let $\mathcal{C}_1$ be any class chosen among* $\mathrm{C_{=}P}$, PP, $\mathrm{P}^{\mathrm{C_{=}P}}$, P^{PP}, *and* PSPACE, *and let $\mathcal{C}_2$ be any class chosen among* $\mathrm{BPP}^{\oplus \mathrm{P}}$, BPP, PH, *and* $\oplus$P. *There exists some oracle A such that $\mathcal{C}_1^A$ contains a $\mathcal{C}_2^A$-immune set.*

What about the converse direction? Does $\text{BPP}^{\oplus\text{P}}$, or even some smaller class, contain a $\doteq\text{P}$-immune, or even a PP-immune, set relative to some oracle? Note that Torán [Tor88, Tor91] provided a simple separation of this kind: There exists an oracle A such that $\text{NP}^A \not\subseteq \doteq\text{P}^A$; see [Bei91] for a simplification of the proof of Torán's result. We strengthen this result by showing that the separation is witnessed by a $\doteq\text{P}^B$-immune set in NP^B for another oracle set B. Indeed, the only property of $\doteq\text{P}$ needed to obtain a relativized separation from NP with immunity is that $\doteq\text{P}$ is closed under finite unions,[3] and this closure property relativizes.

Lemma 7.3.7 *For every oracle A, $\doteq\text{P}^A$ is closed under finite unions. That is, given a finite collection $N_1, N_2, \ldots, N_k$ of NPOTMs, there exists an NPOTM N such that for each input x, N^A accepts x (in the sense of $\doteq\text{P}$) if and only if for some j, N_j^A accepts x (in the sense of $\doteq\text{P}$), i.e., for each $x \in \Sigma^*$,*

$$\text{acc}_{N^A}(x) = \text{rej}_{N^A}(x) \iff (\exists j : 1 \leq j \leq k)\,[\text{acc}_{N_j^A}(x) = \text{rej}_{N_j^A}(x)].$$

Theorem 7.3.8 *There exists some oracle B such that NP^B contains a $\doteq\text{P}^B$-immune set.*

Proof. The witness set here will be L_B, where for any set S,

$$L_S = \{0^n \mid n \geq 1 \text{ and there exists a string of length } n \text{ in } S\}$$

is a set in NP^S. Fix an enumeration $N_1^{(\cdot)}, N_2^{(\cdot)}, \ldots$ of all NPOTMs, again having the property that for infinitely many indices the machine with that index accepts the empty set regardless of the oracle. (Throughout this proof, "acceptance" means "$\doteq\text{P}$ acceptance" as in Lemma 7.3.7.) As in the proof of Theorem 7.3.1, we try to satisfy for each $i \geq 1$ for which N_i^B accepts an infinite subset of L_B, the requirement

$$R_i : \quad L(N_i^B) \cap \overline{L_B} \neq \emptyset.$$

Again, the stage-wise construction of $B = \bigcup_{i \geq 0} B_i$ is initialized by setting B_0 to the empty set and the restraint function t_0 to 0, and we keep a list $\mathcal{L}$ of currently unsatisfied requirements. Stage $i > 0$ is as follows.

Stage i. Add i to $\mathcal{L}$. Consider all machines $N_{\ell_1}^{(\cdot)}, \ldots, N_{\ell_m}^{(\cdot)}$ corresponding to indices ℓ_r that at this point are in $\mathcal{L}$. Let $N_{\mathcal{L}}^{(\cdot)}$ be the machine that exists for $N_{\ell_1}^{(\cdot)}, \ldots, N_{\ell_m}^{(\cdot)}$ by Lemma 7.3.7, i.e., for every oracle Z and for each input x,

$$N_{\mathcal{L}}^Z \text{ accepts } x \iff (\exists r : 1 \leq r \leq m)\,[N_{\ell_r}^Z \text{ accepts } x]. \tag{7.3.4}$$

[3] It is known that $\doteq\text{P}$ is closed even under polynomial-time "positive" Turing reductions, see [LLS75] for the definition. The proof of this closure property of $\doteq\text{P}$ is implicit in the methods of [GNW90], as has been noted in [Rot93] for the positive truth-table case; the same result was noted independently in [BCO93]. We refer to those sources for a proof of Lemma 7.3.7.

Let $p_{\mathcal{L}}$ be the polynomial bounding the runtime of $N_{\mathcal{L}}^{(\cdot)}$. Choose $n = n_i > t_{i-1}$ to be the smallest integer such that $2^n > 2p_{\mathcal{L}}(n)$. Choose an oracle extension $E \subseteq \Sigma^n$ of B_{i-1} such that

$$E = \emptyset \iff N_{\mathcal{L}}^{B_{i-1} \cup E} \text{ accepts } 0^n. \tag{7.3.5}$$

It has been shown in [Bei91] that an oracle extension E satisfying (7.3.5) exists if n is chosen as above. Set B_i to $B_{i-1} \cup E$ and set t_i to $p_{\mathcal{L}}(n)$. If the extension E chosen is the empty set, then by (7.3.5) and (7.3.4), there exists an r, $1 \leq r \leq m$, such that $N_{\ell_r}^{B_{i-1}}$ accepts 0^n. Let $\tilde{r}$ be the smallest such r, and cancel $\ell_{\tilde{r}}$ from $\mathcal{L}$.

End of Stage i.

Note that if we have chosen $E = \emptyset$ in Stage i, then $0^n \notin L_E$ and Requirement $R_{\ell_{\tilde{r}}}$ has been satisfied. On the other hand, if $E \neq \emptyset$, then by (7.3.5) and (7.3.4), we have ensured that (i) $0^n \in L_E$, and (ii) for each r, $1 \leq r \leq m$, $N_{\ell_r}^{B_{i-1} \cup E}$ rejects 0^n. Now, an argument analogous to Claims 2 and 3 in the proof of Theorem 7.3.1 shows that L_B is a $\mathrm{G{=}P}^B$-immune set in NP^B, completing the proof. ∎

Similarly, there exists some oracle C such that NP^C (and thus PH^C and PP^C) has a $\oplus\mathrm{P}^C$-immune set—this result was obtained by Bovet et al. [BCS92], based on their sufficient condition for proving relativized strong separations and on Torán's simple separation of NP and $\oplus$P [Tor91].

Since the inclusions NP $\subseteq$ PP and coNP $\subseteq$ G=P hold relative to every fixed oracle, Theorem 7.3.8 immediately gives the following corollaries.

Corollary 7.3.9 *There exists some oracle B such that* PP^B *contains a* $\mathrm{G{=}P}^B$*-immune set.*

Recall from the introduction that for any complexity class $\mathcal{C}$, a set is said to be *simple* for $\mathcal{C}$ (or $\mathcal{C}$*-simple*) if it belongs to $\mathcal{C}$ and its complement is $\mathcal{C}$-immune. Homer and Maass [HM83] proved the existence of a recursively enumerable set A such that NP^A contains a simple set, and Balcázar [Bal85] improved this result by making A recursive via a novel and very elegant trick: his construction starts with a *full* oracle instead of an empty oracle and then proceeds by *deleting* strings from it. Balcázar's result in turn was generalized by Torenvliet and van Emde Boas [Tor86, TvEB89] to the second level and by Bruschi [Bru92] to all levels of the polynomial hierarchy. Balcázar and Russo [BR88] also proved (relative to some oracle) the existence of a simple set in the one-sided error probabilistic class R, which is contained in NP $\cap$ BPP. Our result below that G=P has a simple set in some relativization (all our oracles are recursive) extends those previous simplicity results that each are restricted to classes contained in the polynomial hierarchy. Since of the classes we consider (PH, PP, $\oplus$P, and G=P), all classes except G=P are known to be closed

under complement, $\mathrm{G{=}P}$ is the only class for which it makes sense to ask about the existence of simple sets.

Corollary 7.3.10 *There exists some oracle B such that $\mathrm{G{=}P}^B$ contains a simple set.*

Proof. Let B be the oracle constructed in the proof of Theorem 7.3.8 and let L_B be the witness set of this proof. Consider the complement $\overline{L_B}$ of L_B in Σ^*. Since $L_B \in \mathrm{NP}^B$, $\overline{L_B}$ is in coNP^B and thus in $\mathrm{G{=}P}^B$. It has been shown in the proof of Theorem 7.3.8 that L_B, the complement of $\overline{L_B}$, is an infinite set having no infinite subset in $\mathrm{G{=}P}^B$. That is, $\overline{L_B}$ is $\mathrm{G{=}P}^B$-simple. ∎

7.4 Immunity Results for Other Counting Classes and Hierarchies

The last section in particular showed that, in suitable relativizations, $\mathrm{G{=}P}$ (and thus PP) is immune to both PH and $\oplus$P (Corollary 7.3.6), and NP (and thus PH and PP) is immune to $\mathrm{G{=}P}$ (Theorem 7.3.8 and Corollary 7.3.9) and to $\oplus$P [BCS92]. In this section, we will prove the existence of oracles relative to which P^{NP} (and thus PH) is immune to PP, and relative to which $\oplus$P is immune to $\mathrm{PP}^{\mathrm{PH}}$. The latter result strengthens the previously known relativized strong separation of $\oplus$P from PH [Ko90] (cf. [Bru92]), and it also implies the new relativized strong separation of $\oplus$P from PP. Noticing that $\mathrm{G{=}P} \subseteq \mathrm{PP}$ holds in all relativizations, we thus have settled all possible relativized strong separation questions involving any pair of classes chosen among PH, PP, $\oplus$P, and $\mathrm{G{=}P}$, as claimed earlier.

We show these remaining results by improving known relativized simple separations to strong ones. The simple separation $(\exists A)\,[\oplus\mathrm{P}^A \not\subseteq \mathrm{PP}^A]$ [Tor88, Tor91] (see also [Bei91]) was strengthened by Green [Gre91] to $(\exists B)\,[\oplus\mathrm{P}^B \not\subseteq \mathrm{PP}^{\mathrm{PH}^B}]$.

Since the analog of Lemma 7.3.7 as well holds for PP (in fact, PP is closed under polynomial-time truth-table reductions [FR91], and this proof relativizes), the following theorem can be shown by the technique used to prove Theorem 7.3.8. First, we state the analog of Lemma 7.3.7 in terms of weak $\mathrm{PP}^{\mathrm{PH}}$ oracle machines. The proof of this lemma simply follows from the relativized version of the proof that PP is closed under finite unions, which is a special case of its closure under truth-table reductions [FR91].

Lemma 7.4.1 *Let A be any oracle and $d \geq 0$ be any integer. Given any finite collection $N_1, N_2, \ldots, N_k$ of weak $\mathrm{PP}^{\mathrm{PH}}$ oracle machines, there exists a weak $\mathrm{PP}^{\mathrm{PH}}$ oracle machine N such that for each input x, N^A accepts x if and only if for some j, $1 \leq j \leq k$, N_j^A accepts x.*

Theorem 7.4.2 *There exists some oracle D such that $\oplus\mathrm{P}^D$ (and thus $\mathrm{P}^{\mathrm{PP}^D}$ and PSPACE^D) contains a $\mathrm{PP}^{\mathrm{PH}^D}$-immune set.*

Proof. Since the proof is very similar to that of Theorem 7.3.8, we only mention the differences. The witness set here will be L_D, where for any set S,

$$L_S = \{0^n \mid n \geq 1 \text{ and there exists an odd number of length } n \text{ strings in } S\}$$

is a set in $\oplus\mathrm{P}^S$. Now, $N_1^{(\cdot)}, N_2^{(\cdot)}, \ldots$ is an enumeration of all weak $\mathrm{PP}^{\mathrm{PH}^{(\cdot)}}$ oracle machines, and "acceptance" refers to such machines. In Stage i of the construction, we again consider all machines $N_{\ell_1}^{(\cdot)}, \ldots, N_{\ell_m}^{(\cdot)}$ corresponding to indices ℓ_r that at this point are in the list $\mathcal{L}$ of currently unsatisfied requirements, and the machine $N_{\mathcal{L}}^{(\cdot)}$ (with polynomial time bound $p_{\mathcal{L}}$) that exists for them by Lemma 7.4.1. Assume $N_{\mathcal{L}}^{(\cdot)}$ is a $\mathrm{PP}^{\Sigma_d^{p,(\cdot)}}$ machine, and let c_d be the constant that exists for such machines by [Gre91, Thm. 5]. Then, as shown in [Gre91, Thm. 7], choosing $n = n_i > t_{i-1}$ to be the smallest integer such that

$$2p_{\mathcal{L}}(n) \leq \min\{(2^n)^{1/d^2}, c_d 2^{n(d+1)/d^2} - 1\}$$

implies that there exists an extension $E \subseteq \Sigma^n$ of the oracle as constructed so far, D_{i-1}, such that $0^n \in L_E$ if and only if $N_{\mathcal{L}}^{D_{i-1} \cup E}$ rejects 0^n. ∎

Corollary 7.4.3 *There exists some oracle D such that $\oplus\mathrm{P}^D$ contains a set immune to PP^D and to PH^D.*

By essentially the same arguments, also the very recent result of Berg and Ulfberg [BU] that there is an oracle relative to which the levels of the $\mathrm{PP}^{\mathrm{PH}} = \bigcup_{d \geq 0} \mathrm{PP}^{\Sigma_d^p}$ hierarchy separate can be strengthened to level-wise strong separations of this hierarchy. Note that this result generalizes Beigel's [Bei94] result that $(\exists A)\,[\mathrm{P}^{\mathrm{NP}^A} \not\subseteq \mathrm{PP}^A]$. The proof of Theorem 7.4.4 is omitted, since it is very similar to the previous proofs, the only difference being that it is based on the construction given in [BU]. The interested reader is referred to [Rot98c] for a complete proof of this result.

Theorem 7.4.4 *For any $d \geq 1$, there exists some oracle F such that $\mathrm{P}^{\Sigma_d^{p,F}}$ contains a $\mathrm{PP}^{\Sigma_{d-1}^{p,F}}$-immune set. In particular, $\mathrm{P}^{\mathrm{NP}^F}$ (and thus PH^F) has a PP^F-immune set.*

7.5 Conclusions and Open Problems

In this chapter, we have shown that all possible relativized separations involving the polynomial hierarchy and the counting classes C=P, PP, and $\oplus$P can be made strong. In

particular, we have extended to these counting classes previously known strong separations of Ko [Ko90] and Bruschi [Bru92], and we have strengthened to strong separations previously known simple separations of Torán [Tor88, Tor91], Green [Gre91], and Berg and Ulfberg [BU]. We have also shown that $\mathrm{G{=}P}$ contains a simple set relative to some oracle, complementing the corresponding results of Balcázar and Russo [Bal85, BR88] for NP and R, and of Torenvliet and van Emde Boas [Tor86, TvEB89] and Bruschi [Bru92] for Σ_k^p, $k > 1$. However, many questions remain open. The most obvious question is whether these immunity results can be strengthened to bi-immunity or even to balanced immunity; see, e.g., the paper [HZ96].

Regarding the existence of simple sets in $\mathrm{G{=}P}^B$, note that our construction of B can easily be interleaved with other immunity oracle constructions to show results such as: There exists an oracle A such that $\mathrm{G{=}P}^A$ contains a simple set and another set that is P^A-immune; see [Bal85] for the analogous result for NP. Torenvliet and van Emde Boas [Tor86, TvEB89] have even constructed an oracle relative to which NP contains a language that *simultaneously* is simple and P-immune. Can this also be shown to hold for $\mathrm{G{=}P}$?

Our main result that there exists some A such that $\mathrm{G{=}P}^A$ contains a $\mathrm{BPP}^{\oplus \mathrm{P}^A}$-immune set is optimal in the sense that for all oracles B, $\mathrm{G{=}P}^B$ clearly is contained in PP^B and thus in $\mathrm{PP}^{\oplus \mathrm{P}^B}$. However, it is also known that $\mathrm{BPP}^{\oplus \mathrm{P}} \subseteq \mathrm{Almost}[\oplus \mathrm{P}]$ [TO92, RR95], where for any relativized class $\mathcal{C}$, $\mathrm{Almost}[\mathcal{C}]$ denotes the class of languages L such that for almost all oracle sets X, L is in $\mathcal{C}^X$ [NW94]. It is an open problem whether $\mathrm{BPP}^{\oplus \mathrm{P}} = \mathrm{Almost}[\oplus \mathrm{P}]$, see the paper [RR95]. So it is possible that $\mathrm{Almost}[\oplus \mathrm{P}]$ is a strictly larger class than $\mathrm{BPP}^{\oplus \mathrm{P}}$. It is unlikely that $\mathrm{G{=}P}$ is contained in $\mathrm{Almost}[\oplus \mathrm{P}]$. Is there an oracle relative to which $\mathrm{G{=}P}$ is even immune to $\mathrm{Almost}[\oplus \mathrm{P}]$? We conjecture that this is the case. Relatedly, can any of the immunity results of this chapter be shown to hold with probability 1 relative to a random oracle?

Chapter 8

Tally NP Sets and Easy Census Functions

8.1 Introduction

Does every P set have an easy (i.e., polynomial-time computable) census function? Many important properties similar to this one were studied during the past decades to gain insight into the nature of feasible computation. Among the questions that were previously studied are the question of whether or not every P set has an easy-to-compute ranking function [GS91, HR90], whether every P set is P-isomorphic to some rankable set [GH96], whether every sparse set in P is P-printable [HY84, AR88, RRW94], whether every infinite set in P has an infinite P-printable subset [AR88] (note also the results of Chapter 3, in particular Theorem 3.3.3), whether every P-printable set is P-isomorphic to some tally set in P [AR88], and whether every P set admits easy certificate schemes (see Chapter 3), to name just a few. Some of those questions arise in the field of data compression and are related to Kolmogorov complexity, some are linked to the question of whether one-way functions exist.

Extending this line of research, the present chapter studies the complexity of computing the census functions of sets in P. Census functions have proven to be a particularly important and useful notion in complexity theory, and their use has had a profound impact upon almost every area of the field. In particular, consider the extensive literature related to the Isomorphism Conjecture of Berman and Hartmanis (e.g., [BH77, Mah82], and many other papers), the work on the existence of Turing-hard sparse sets (or of polynomial-size circuits) for various complexity classes (e.g., [KL80, KS85, BBS86, HR97b]), the results relating the computation times for NP sets to their densities and the results on P-printability [HY84, AR88, RRW94, GH96], the upward separation technique (e.g.,

[Har83b, HIS85, All91, RRW94, HJ95], see [HHH99] for more recent advances that are not based on census functions), the results on positive relativization and relativization to sparse oracles (e.g., [Lon85, LS86, BBS86]), the unexpected collapse of the strong exponential-time hierarchy [Hem89], and applications to extended lowness [HJRW98].

Valiant, in his seminal papers [Val79a, Val79b], introduced #P, the class of functions that count the solutions of NP problems, and its tally version $\#\mathrm{P}_1$ for which the inputs are given in unary. Although $\#\mathrm{P}_1$ has not become as prominent as #P, it contains a number of quite interesting and important problems such as the problem `Self-Avoiding Walk` (see [Wel93]): Given an integer n in unary, compute the number of self-avoiding walks on the square lattice having length n and rooted at the origin. `Self-Avoiding Walk` is a well-known classical problem of statistical physics and polymer chemistry, and it is an intriguing open question whether `Self-Avoiding Walk` is $\#\mathrm{P}_1$-complete, see [Wel93]. Known problems complete for $\#\mathrm{P}_1$ [Val79b] have the form: Given an integer n in unary, compute the number of graphs having n vertices and satisfying a fixed graph property π.

In Section 8.3, we will characterize the question of whether every P set has an easy census function in terms of collapses of language and function classes that are considered to be unlikely. In particular, every P set has an easy census function if and only if $\#\mathrm{P}_1 \subseteq \mathrm{FP}$. The main technical contribution in Section 8.3 is Theorem 8.3.7: $\#\mathrm{P}_1^{\mathrm{PH}}$ is contained in $\mathrm{FP}^{\#\mathrm{P}_1^{\#\mathrm{P}_1}}$. An immediate consequence of this result are upward collapse results of the form: The collapse $\#_1 \cdot \mathrm{P} \subseteq \mathrm{FP}$ implies the collapse $\#_1 \cdot \mathrm{PH} \subseteq \mathrm{FP}$. Thus, every P set has an easy census function if and only if every set in the polynomial hierarchy has an easy census function. Note that the corresponding upward collapse for the # operator applied to the levels of PH follows immediately from the upward collapse property of the polynomial hierarchy itself: $\# \cdot \mathrm{P} \subseteq \mathrm{FP}$ implies $\mathrm{NP} = \mathrm{P}$ and thus $\mathrm{PH} = \mathrm{P}$; so, $\# \cdot \mathrm{PH} = \# \cdot \mathrm{P} \subseteq \mathrm{FP}$. However, for the $\#_1$ operator this is not so clear, since the assumption $\#_1 \cdot \mathrm{P} \subseteq \mathrm{FP}$ merely implies that all *tally* NP sets are in P (equivalently, $\mathrm{NE} = \mathrm{E}$), from which one cannot immediately conclude that $\#_1 \cdot \mathrm{PH}$ or even $\#_1 \cdot \mathrm{NP}$ is contained in FP. In fact, Hartmanis, Immerman, and Sewelson [HIS85] show that in some relativized world, $\mathrm{NE} = \mathrm{E}$ and yet the (weak) exponential-time hierarchy does not collapse. In light of this result, it is quite possible that the assumption of all tally NP sets being in P does not force all tally sets from higher levels of the polynomial hierarchy into P.

We show that the assumption $\#\mathrm{P}_1 \subseteq \mathrm{FP}$ implies both $\mathrm{P} = \mathrm{BPP}$ and $\mathrm{PH} \subseteq \mathrm{MOD}_k\mathrm{P}$ for each $k \geq 2$ (Theorem 8.3.6). We also relate a set's property of having an easy census function to other well-studied properties of sets, such as rankability [GS91] and scalability [GH96]. In particular, though every rankable set has an easy census function, we show that (even when restricted to the sets in P) the converse is not true unless $\mathrm{P} = \mathrm{PP}$. This expands the result of Hemaspaandra and Rudich that every P set is rankable if and only if $\mathrm{P} = \mathrm{PP}$ [HR90] by showing that $\mathrm{P} = \mathrm{PP}$ is already implied by the apparently weaker

hypothesis that every P set *with an easy census function* is rankable.

Cai and Hemaspaandra [CH89] introduced the notion of enumerative counting as a way of approximating the value of a #P function deterministically in polynomial time. Hemaspaandra and Rudich [HR90] show that every P set is k-enumeratively rankable for some fixed k in polynomial time if and only if #P = FP. They conclude that it is no more likely that one can enumeratively rank all sets in P than that one can exactly compute their ranking functions in polynomial time. In Section 8.4, we similarly characterize the question of whether every P set has a census function that is n^α-enumerable in time n^β for fixed constants α and β (equivalently, whether every $\#\mathrm{P}_1$ function is n^α-enumerable in time n^β). We show that this hypothesis implies $\#\mathrm{P}_1 \subseteq$ FP, and we conclude that it is no more likely that one can n^α-enumerate the census function of every P set in time n^β than that one can precisely compute its census function in polynomial time.

Finally, Section 8.5 provides a number of relativization results.

8.2 Notation and Definitions

For any set L, the *census function* of L, $\mathit{census}_L : \Sigma^* \to \mathbb{N}$, is defined by

$$\mathit{census}_L(1^n) = ||L^{=n}||.$$

We note that the census function of L at n is often defined as the number of elements in L of length up to n in the literature. This definition and our definition are compatible as long as our computability admits subtraction. We also note that we let census_L map strings 1^n (as opposed to numbers n in binary notation) to $||L^{=n}||$ to emphasize that the input to the transducer computing census_L is given in unary.

The definitions of *sparse* sets and *tally* sets are given in Chapter 2 on page 17. Recall that FP is the class of polynomial-time computable functions. We write FP_1 to denote the class of functions computable in polynomial time by deterministic transducers with a unary input alphabet. Recall that an unambiguous Turing machine is a nondeterministic Turing machine that on each input has at most one accepting path. UE, the exponential-time analog of the class UP (defined in Chapter 2), is the class of all languages accepted by some unambiguous Turing machine running in in time 2^{cn} for some constant c.

Recall that for any nondeterministic Turing machine M and any input $x \in \Sigma^*$, $\mathrm{acc}_M(x)$ denotes the number of accepting paths of $M(x)$. A *spanP machine* [KST89] is an NP machine that has a special output device on which some output is printed for each accepting path. For any spanP machine M and any input $x \in \Sigma^*$, $\mathrm{span}_M(x)$ is defined to be the number of distinct outputs of $M(x)$ if $M(x)$ has at least one accepting path, and 0 otherwise.

A *tally NP machine* (respectively, a *tally spanP machine*) is an NP (respectively, a spanP) machine with a unary input alphabet.

Recall from Chapter 2 the definition of the function class

$$\#\mathrm{P} = \{\mathrm{acc}_M \mid M \text{ is an NP machine}\}.$$

A number of related function classes are defined as follows.

Definition 8.2.1 *1.* [Val79b] $\#\mathrm{P}_1 = \{\mathrm{acc}_M \mid M$ *is a tally* NP *machine*$\}$.

2. [KST89] $\mathrm{spanP} = \{\mathrm{span}_M \mid M$ *is a* spanP *machine*$\}$.

3. $\mathrm{spanP}_1 = \{\mathrm{span}_M \mid M$ *is a tally* spanP *machine*$\}$.

4. $\#\mathrm{E} = \{\mathrm{acc}_M \mid M$ *is an* NE *machine*$\}$.

Recall the notion of the "#" operator provided by Definition 2.2.2 on page 15 that generalizes the class #P. The following definition gives the "tally" analog of this operator, which generalizes the class $\#\mathrm{P}_1$.

Definition 8.2.2 *For any language class $\mathcal{C}$, define $\#_1 \cdot \mathcal{C}$ to be the class of functions $f : \Sigma^* \to \mathbb{N}$ for which there exist a set $A \in \mathcal{C}$ and a polynomial p such that for each $n \in \mathbb{N}$,*

$$f(1^n) = ||\{y \mid |y| = p(n) \text{ and } \langle 1^n, y\rangle \in A\}||.$$

As stated below, both operators, # and $\#_1$, are monotonic. Since this property follows immediately from the definitions, we omit the proof of the following proposition.

Proposition 8.2.3 *Let $\mathcal{C}$ and $\mathcal{D}$ be any classes of sets.*

1. If $\mathcal{C} \subseteq \mathcal{D}$, then $\# \cdot \mathcal{C} \subseteq \# \cdot \mathcal{D}$.

2. If $\mathcal{C} \subseteq \mathcal{D}$, then $\#_1 \cdot \mathcal{C} \subseteq \#_1 \cdot \mathcal{D}$.

Next, we gather some easy observations regarding equivalent formulations of the classes $\#\mathrm{P}^{\mathrm{PH}}$ and $\#\mathrm{P}_1^{\mathrm{PH}}$ to be used in Section 8.3. Analogs of Proposition 8.2.4 for classes other than PH as the oracle class could be stated as well; we focus here on the class of interest to us.

Recall that, for any language class $\mathcal{C}$, we write $\mathrm{P}^{\mathcal{C}[1]}$ to indicate that on every input in the $\mathrm{P}^{\mathcal{C}}$ computation at most one call to the $\mathcal{C}$ oracle is allowed. Similarly, for any function class $\mathcal{F}$, we write $\mathrm{P}^{\mathcal{F}[1]}$ to indicate that on every input in the $\mathrm{P}^{\mathcal{F}}$ computation at most one call to the function oracle from $\mathcal{F}$ is allowed.

Proposition 8.2.4 *The following three statements are true.*

1. $\mathrm{P}^{\mathrm{PH}} = \mathrm{P}^{\mathrm{PH}[1]}$.
2. $\#_1 \cdot \mathrm{P}^{\mathrm{PH}} = \#_1 \cdot \mathrm{P}^{\mathrm{PH}[1]} = \#_1 \cdot \mathrm{PH} = \#\mathrm{P}_1^{\mathrm{PH}[1]} = \#\mathrm{P}_1^{\mathrm{PH}}$.
3. $\# \cdot \mathrm{P}^{\mathrm{PH}} = \# \cdot \mathrm{P}^{\mathrm{PH}[1]} = \# \cdot \mathrm{PH} = \#\mathrm{P}^{\mathrm{PH}[1]} = \#\mathrm{P}^{\mathrm{PH}}$.

Proof. Part 1 follows immediately from the fact that PH is closed under polynomial-time Turing reductions. That is,

$$\mathrm{P}^{\mathrm{PH}} \subseteq \mathrm{PH} \subseteq \mathrm{P}^{\mathrm{PH}[1]} \subseteq \mathrm{P}^{\mathrm{PH}};$$

so, the above inclusions are equalities.

Part 2: From the proof of part 1 and the monotonicity of the $\#_1$ operator (see Proposition 8.2.3), we have the first two equalities: $\#_1 \cdot \mathrm{P}^{\mathrm{PH}} = \#_1 \cdot \mathrm{P}^{\mathrm{PH}[1]} = \#_1 \cdot \mathrm{PH}$.

To see that $\#_1 \cdot \mathrm{PH} \subseteq \#\mathrm{P}_1^{\mathrm{PH}[1]}$, let $f \in \#_1 \cdot \mathrm{PH}$ be witnessed by a set $A \in \mathrm{PH}$ and a polynomial p; i.e., for each $n \in \mathbb{N}$,

$$f(1^n) = ||\{y \mid |y| = p(n) \text{ and } \langle 1^n, y\rangle \in A\}||.$$

Consider the following tally NP oracle machine M. On input 1^n, M with oracle A guesses a string y of length $p(n)$, and for each y guessed, M accepts if and only if $\langle 1^n, y\rangle \in A$. Hence, $f \in \#\mathrm{P}_1^{\mathrm{PH}[1]}$.

Since $\#\mathrm{P}_1^{\mathrm{PH}[1]} \subseteq \#\mathrm{P}_1^{\mathrm{PH}}$, it remains to show that $\#\mathrm{P}_1^{\mathrm{PH}} \subseteq \#_1 \cdot \mathrm{PH}$. Let $f \in \#\mathrm{P}_1^{\mathrm{PH}}$ be witnessed by a tally NP oracle machine M with oracle $A \in \mathrm{PH}$; i.e., $f = \mathrm{acc}_{M^A}$. We assume that all computation paths of M on input 1^n are encoded as strings in $\{0,1\}^{p(n)}$ for some polynomial p, where the oracle queries that are asked on such a path and the corresponding answers are part of the encoding string. Define B to be the set of all strings $\langle 1^n, y\rangle$ such that

- $n \in \mathbb{N}$,
- $y \in \{0,1\}^{p(n)}$ encodes an accepting computation path of $M^A(1^n)$ with oracle queries $q_1, q_2, \ldots, q_k$, and
- for each i with $1 \leq i \leq k$, $M^A(1^n)$ on path y proceeds in the "yes" state if and only if $q_i \in A$.

It follows that $B \in \mathrm{PH}$ and that for each $n \in \mathbb{N}$,

$$f(1^n) = ||\{y \mid |y| = p(n) \text{ and } \langle 1^n, y\rangle \in A\}||.$$

Hence, $f \in \#_1 \cdot \mathrm{PH}$, completing the proof of part 2.

The proof of part 3 is analogous to the proof of part 2. ∎

Recall the notion of P-isomorphism from Definition 2.2.6 on page 16. We now define two special types of P-isomorphisms, the length-preserving and the order-preserving P-isomorphisms.

Definition 8.2.5

1. *A* P*-isomorphism ϕ is* length-preserving *if for all $x \in \Sigma^*$, $|\phi(x)| = |x|$.*
2. *A* P*-isomorphism ϕ mapping set $A \subseteq \Sigma^*$ to set $B \subseteq \Sigma^*$ is* order-preserving *if for any two strings x and y satisfying either $x, y \in A$ or $x, y \notin A$, if $x \leq y$ then $\phi(x) \leq \phi(y)$.*

The notion of rankability is given in Definition 2.2.7 on page 17. Goldsmith and Homer [GH96] introduced the property of scalability, a more flexible notion than rankability in which the rank of some given element within the set is not necessarily determined with respect to the lexicographic order of Σ^*, but rather with respect to *any* well-ordering of Σ^* that can be "scaled" by a polynomial-time computable and polynomial-time invertible bijection between $\mathbb{N}$ and Σ^*. Equivalently, Goldsmith and Homer [GH96] proved that the scalable sets are precisely those that are P-isomorphic to some rankable set. The definition below is based on this characterization.

Definition 8.2.6 [GH96] *A language A is* scalable *if it is P-isomorphic to a rankable set. For any oracle X, the X-*scalable *sets are those that are* P^X*-isomorphic to some set rankable in* FP^X.

8.3 Does P Have Easy Census Functions?

We start by exploring the relationships between the properties of a set being rankable, being scalable, and having an easy census function. Let A be any set (not necessarily in P). Consider the following conditions:

(i) A is rankable.

(ii) A has an easy census function.

(iii) A is P-isomorphic to some rankable set (i.e., A is scalable).

(iv) A is P-isomorphic to some rankable set via some length-preserving isomorphism.

(v) A is P-isomorphic to some rankable set via some order-preserving isomorphism.

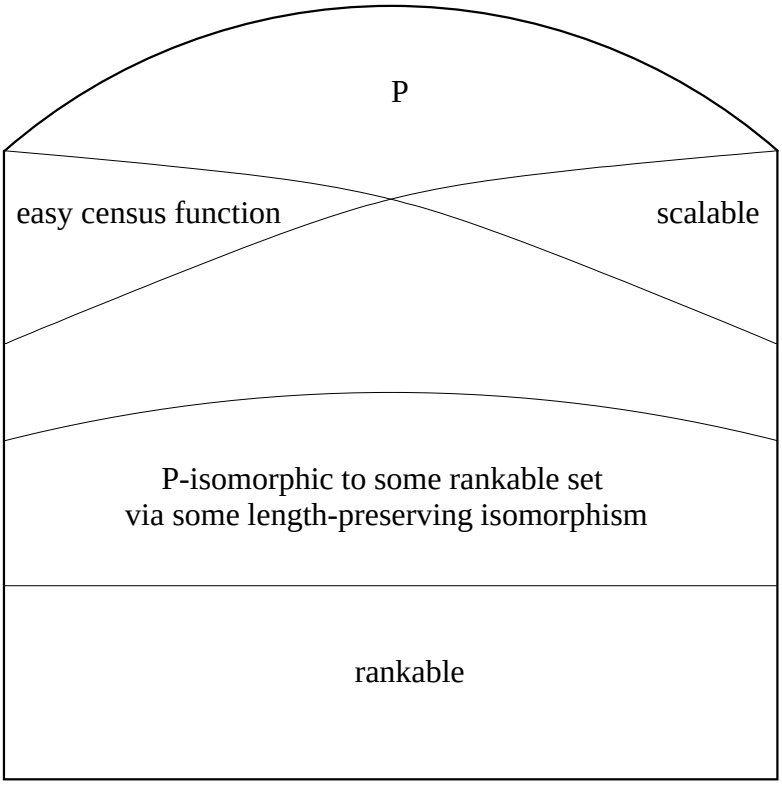

Figure 8.1: Inclusion structure of the sets in P satisfying Properties (i) through (iv)

It is immediately clear that for any set A, (i) implies each of (ii), (iv), and (v), and each of (iv) and (v) implies (iii). The next proposition shows that the rankable sets are closed under order-preserving P-isomorphisms (thus, conditions (i) and (v) in fact are equivalent) and that the class of sets having an easy census function is closed under length-preserving P-isomorphisms. The latter fact immediately gives that (iv) implies (ii), since every rankable set has an easy census function. The inclusion structure of the sets in P satisfying Properties (i) through (iv) is given in Figure 8.1.

Proposition 8.3.1 *The following two statements are true.*

1. *The class of all rankable sets is closed under order-preserving* P*-isomorphisms.*
2. *The class of sets having an* FP*-computable census function is closed under length-preserving* P*-isomorphisms.*

Proof. (1) Let A be P-isomorphic to a rankable set, B, via some order-preserving isomorphism, ϕ. Since B is rankable, $\overline{B}$ is rankable. Let respectively r and $\bar{r}$ be the ranking

functions for B and $\overline{B}$. For any string $x \in \Sigma^*$, let lex(x) denote the lexicographic order of x; i.e., the number of strings $w \in \Sigma^*$ with $w \leq x$. Define the function

$$r'(x) = \begin{cases} r(\phi(x)) & \text{if } x \in A \\ \text{lex}(x) - \bar{r}(\phi(x)) & \text{if } x \notin A. \end{cases}$$

Clearly, r' is computable in polynomial time and r' is the ranking function for A.

(2) Let A be P-isomorphic to a set B with *census*$_B \in$ FP via some length-preserving isomorphism, ϕ. Then, for each n, $\phi(A^{=n}) = B^{=n}$. Thus, *census*$_A = census_B$, which implies *census*$_A \in$ FP. ∎

So we are left with only the four conditions (i) to (iv). Since there are nonrecursive sets with an FP-computable census function, but any set satisfying one of (i), (iii), or (iv) is in P, condition (ii) in general cannot imply any of the other three conditions. On the other hand, when we restrict our attention to the sets in P having easy census functions, we can show that (ii) implies (i) if and only if P = PP. Thus, even when restricted to P sets, it is unlikely that (ii) is equivalent to (i).

Theorem 8.3.2 *All* P *sets with an easy census function are rankable if and only if* P = PP.

Proof. Hemaspaandra and Rudich [HR90] show that $\text{P}^{\#\text{P}} = \text{P}$ if and only if every P set is rankable. Noticing that P = PP is equivalent to $\text{P}^{\#\text{P}} = \text{P}$, this result in particular implies that every P set with an easy census function is rankable if P = PP.

Conversely, assume that every P set with an easy census function is rankable. We show that this assumption implies P = PP. Let L be any set in PP, and let A be a set in P and p be a polynomial such that for all $x \in \Sigma^*$,

$$x \in L \iff ||\{y \mid |y| = p(|x|) \text{ and } x\#y \in A\}|| \geq 2^{p(|x|)-1}.$$

Define

$$T = \{b\#x\#y \mid x, y \in \Sigma^*,\ |y| = p(|x|),\ b \in \{0,1\},\ \text{and } \chi_A(x\#y) = b\}.$$

Clearly, $T \in$ P. Also, the census function of T is easy to compute: Given n in unary, compute the largest integer i such that $i + p(i) + 3 \leq n$. Then,

$$\textit{census}_T(1^n) = \begin{cases} 2^{i+p(i)} & \text{if } i + p(i) + 3 = n \\ 0 & \text{if } i + p(i) + 3 < n. \end{cases}$$

Since $T \in$ P and *census*$_T \in$ FP, our hypothesis implies that T is rankable. Let r be the ranking function for T. For each $x \in \Sigma^+$, let $\widehat{x}$ denote the lexicographic predecessor of x.

Note that, for each $x \in \Sigma^+$, $r(0\#x\#1^{p(|x|)}) - r(1\#\widehat{x}\#1^{p(|\widehat{x}|)})$ gives the number of strings y of length $p(|x|)$ such that $x\#y \notin A$. Hence, for each $x \in \Sigma^+$,

$$x \in L \iff r(0\#x\#1^{p(|x|)}) - r(1\#\widehat{x}\#1^{p(|\widehat{x}|)}) \leq 2^{p(|x|)-1}.$$

Since the predicate on the right-hand side of the above equivalence can be decided in polynomial time, it follows that $L \in$ P. ∎

Corollary 8.3.3 *All* P *sets are rankable if and only if all sets in* P *with an easy census function are rankable.*

One might ask whether or not all P sets outright have an easy census function (which, if true, would make Corollary 8.3.3 trivial). The following characterization of this question in terms of unlikely collapses of certain function and language classes suggests that this probably is not true. Thus, Corollary 8.3.3 is nontrivial with the same certainty with which we believe that for instance not all $\#\mathrm{P}_1$ functions are in FP.[1]

Theorem 8.3.4 *The following five statements are equivalent.*

1. *Every* P *set has an* FP*-computable census function.*
2. $\#\mathrm{P}_1 \subseteq \mathrm{FP}$.
3. $\#\mathrm{E} = \mathrm{FE}$.
4. $\mathrm{P}^{\#\mathrm{P}_1} = \mathrm{P}$.
5. *For every language L accepted by a logspace-uniform depth* 2 AND-OR *circuit family of bottom fan-in* 2*, census_L is in* FP.

Proof. To show that (1) implies (2), let f be any function in $\#\mathrm{P}_1$. Let M be some tally NP machine with $\mathrm{acc}_M = f$. Assume that M runs in time n^k, for some constant k. Define

$$A = \{x \mid |x| = n^k \text{ for some } n \text{ and } x \text{ encodes an accepting path of } M(1^n)\}.$$

Clearly, A is in P (note that n can be found in polynomial time, since computing the kth root of some integer can be done in polynomial time). Now from our hypothesis it follows that census_A is in FP, and since $\mathit{census}_A = \mathrm{acc}_M$, we have $f \in$ FP.

[1] It is not difficult to construct—by standard techniques—an oracle relative to which $\#\mathrm{P}_1 \not\subseteq$ FP. On the other hand, we will show in Section 8.5 that, relative to some oracle, $\#\mathrm{P}_1 \subseteq$ FP, yet $\#\mathrm{P} \neq$ FP (and thus PP $\neq$ P).

Conversely, let A be an arbitrary set in P. Define M to be the tally NP machine that, on input 1^n, guesses an $x \in \{0,1\}^n$, and for each x guessed, accepts along the path for x if and only if $x \in A$. Then, $\text{acc}_M = \textit{census}_A$. Since by hypothesis $\text{acc}_M \in$ FP, it follows that $\textit{census}_A \in$ FP.

The equivalence of (2) and (3) can be proven by means of standard translation—this is essentially the function analog of Book's result that every tally NP set is in P if and only if NE = E [Boo74]; see the papers [Har83b, HIS85] for the extension of this result to sparse sets.

The equivalence of (2) and (4) is straightforward.

It is easy to see that (2) implies (5). In order to prove that (5) implies (2), note that computing the number of satisfying assignments for monotone 2CNF formulas is complete for #P [Val79b] under logspace reductions. Now, given a function f in $\#\text{P}_1$, there exist logspace-computable functions R, S, and ρ such that for all n, $R(1^n)$ is a monotone 2CNF formula with $\rho(1^n)$ variables, and $f(1^n)$ equals the number of satisfying assignments for $R(1^n)$ divided by $S(1^n)$. The reduction R can be modified so that for every n, $\rho(1^{n+1}) > \rho(1^n)$. Now let C_m be the circuit defined as follows: (a) if $m = \rho(1^n)$ for some n, then C_m is a depth 2 AND-OR circuit that tests whether an assignment, given as the input, satisfies $R(1^n)$, and (b) if not, C_m is a depth 1 AND circuit that rejects all inputs. This circuit family $F = \{C_m\}$ is logspace-uniform. Now let A be the language accepted by F. Then, for every n, $f(1^n) = \textit{census}_A(1^{\rho(1^n)})/S(1^n)$. Thus, (5) implies that $f \in$ FP. ∎

Theorem 8.3.4 can as well be stated for more general classes than $\#\text{P}_1 = \#_1 \cdot \text{P}$. In particular, this comment applies to $\#_1 \cdot \mathcal{C}$, where for instance $\mathcal{C} =$ NP or $\mathcal{C} =$ PH. Noticing that $\text{spanP}_1 = \#_1 \cdot \text{NP}$ and focusing on the first two conditions of Theorem 8.3.4, this observation is exemplified as follows.

Theorem 8.3.5 *The following two statements are true.*

1. *Every* NP *set has an* FP*-computable census function if and only if* $\text{spanP}_1 \subseteq$ FP.
2. *Every set in* PH *has an* FP*-computable census function if and only if* $\#_1 \cdot \text{PH} \subseteq$ FP.

We will show later that the conditions of Theorem 8.3.4 in fact are equivalent to the two conditions stated in either part of Theorem 8.3.5.

We now give two more consequences of the assumption $\#\text{P}_1 \subseteq$ FP.

Theorem 8.3.6 *If* $\#\text{P}_1 \subseteq$ FP*, then the following two statements are true.*

1. *For any fixed* $k \geq 2$, $\text{PH} \subseteq \text{MOD}_k\text{P}$, *and*
2. $\text{P} = \text{BPP}$.

Proof. Suppose $\#\mathrm{P}_1 \subseteq \mathrm{FP}$. In order to prove the first part, note that if a natural number $k \geq 2$ has prime factorization of the form $p_1^{e_1} \cdots p_t^{e_t}$, then $\mathrm{MOD}_k\mathrm{P} = \{L_1 \cap \cdots \cap L_t \mid L_1 \in \mathrm{MOD}_{p_1}\mathrm{P}, \ldots, L_t \in \mathrm{MOD}_{p_t}\mathrm{P}\}$ [Her90, BG92]. Thus it suffices to show that for every prime $k \geq 2$, $\mathrm{PH} \subseteq \mathrm{MOD}_k\mathrm{P}$.

We claim that for every prime $k \geq 2$, each language in PH belongs to $\mathrm{MOD}_k\mathrm{P}/\mathrm{poly}$ with an advice function in $\mathrm{FP}^{\#\mathrm{P}[1]}$. To prove the claim, let L be any language in PH and $k \geq 2$ be any prime number. Toda and Ogihara [TO92] prove that some $A \in \mathrm{MOD}_k\mathrm{P}$ witnesses that $L \in \mathrm{MOD}_k\mathrm{P}/\mathrm{poly}$ together with some polynomially length-bounded advice function. Fix such an A and define

$$B = \{\langle 1^n, w\rangle \mid n \geq 1 \text{ and for every } x, |x| = n, x \in L \text{ if and only if } \langle x, w\rangle \in A\}.$$

Define f to be the function that, for each n, maps 1^n to the lexicographically smallest string w such that $\langle 1^n, w\rangle \in B$. Since $L \in \mathrm{PH}$ and $A \in \mathrm{MOD}_k\mathrm{P}$, B belongs to $\mathrm{coNP}^{\mathrm{PH} \cup \mathrm{MOD}_k\mathrm{P}}$, which is included in $\mathrm{PH}^{\mathrm{MOD}_k\mathrm{P}}$. Then f is total and $f \in \mathrm{FP}^{\mathrm{NP}^B} \subseteq \mathrm{FP}^{\mathrm{PH}^{\mathrm{MOD}_k\mathrm{P}}}$. Toda and Ogihara show that $\mathrm{PH}^{\mathrm{MOD}_k\mathrm{P}} \subseteq \mathrm{BPP}^{\mathrm{MOD}_k\mathrm{P}}$. A part of the proof of Toda's Theorem [Tod91b] shows $\mathrm{BPP}^{\oplus\mathrm{P}} \subseteq \mathrm{P}^{\#\mathrm{P}[1]}$. By following the same argument one can show that for every prime $k \geq 2$, $\mathrm{BPP}^{\mathrm{MOD}_k\mathrm{P}} \subseteq \mathrm{P}^{\#\mathrm{P}[1]}$, completing the proof of the claim.

Since f can be computed in $\mathrm{FP}^{\#\mathrm{P}[1]}$, the set

$$\{\langle 1^n, i, b\rangle \mid n \geq 1, b \in \{0,1\}, \text{ and the } i\text{th bit of } f(1^n) \text{ is } b\}$$

can be decided in polynomial time with one query to a suitable function h in $\#\mathrm{P}$. Let $q_{n,i}$ be the string that is queried on input $\langle 1^n, i, b\rangle$. Define a $\#\mathrm{P}_1$ function g by

$$g(1^n) = \langle h(q_{n,1}), h(q_{n,2}), \ldots, h(q_{n,p(n)})\rangle,$$

where p is a polynomial bounding the length of the advice and the value of $g(1^n)$ is viewed as a number written in binary. One query to g will allow us to compute f; i.e., $f \in \mathrm{FP}_1^{\#\mathrm{P}_1[1]}$.

Applying our supposition $\#\mathrm{P}_1 \subseteq \mathrm{FP}$, we conclude that f can be computed in polynomial time. Since L is in $\mathrm{MOD}_k\mathrm{P}/\mathrm{poly}$ with polynomial-time computable advice, it follows that $L \in \mathrm{MOD}_k\mathrm{P}$. Hence, $\mathrm{PH} \subseteq \mathrm{MOD}_k\mathrm{P}$.

In order to prove the second part, note that $\mathrm{BPP} \subseteq \mathrm{P}/\mathrm{poly}$ [Adl78] and $\mathrm{BPP} \subseteq \mathrm{PH}$ [Sip83b, Lau83]. By following the proof of the first part with P in place of $\mathrm{MOD}_k\mathrm{P}$ we obtain that $\mathrm{BPP} \subseteq \mathrm{P}$. ∎

Now we show that the conditions of Theorem 8.3.4 in fact are equivalent to the two conditions stated in either part of Theorem 8.3.5. To this end, we establish the following theorem, which is interesting in its own right. Theorem 8.3.7 is the main technical contribution in this section.

Theorem 8.3.7 $\#\mathrm{P}_1{}^{\mathrm{PH}} \subseteq \mathrm{FP}^{\#\mathrm{P}_1{}^{\#\mathrm{P}_1}}$.

Before we turn to the proof of Theorem 8.3.7, we discuss some issues related to this result.

First, we stress that Theorem 8.3.7 is a novel insight and does not trivially follow from known results. In particular, Toda's result that $\mathrm{PH} \subseteq \mathrm{P}^{\#\mathrm{P}[1]}$ does not imply Theorem 8.3.7 in any obvious way. Note that Toda's Theorem does imply the following two inclusions:[2]

$$\#\mathrm{P}_1{}^{\mathrm{PH}} \subseteq \#\mathrm{P}_1{}^{\#\mathrm{P}[1]}, \text{ and} \tag{8.3.1}$$

$$\#\mathrm{P}^{\mathrm{PH}} \subseteq \#\mathrm{P}^{\#\mathrm{P}[1]}. \tag{8.3.2}$$

Observe, however, that the oracles on the right-hand sides of the inclusions (8.3.1) and (8.3.2) are #P functions. In contrast, Theorem 8.3.7 establishes containment of $\#\mathrm{P}_1{}^{\mathrm{PH}}$ in a class in which only $\#\mathrm{P}_1$ oracles occur. Although our proof also applies the techniques of Toda [Tod91b] and Toda and Ogihara [TO92], our *result* seems to be incomparable with the above consequence (8.3.1) of Toda's Theorem.

Second, can Theorem 8.3.7 be strengthened to $\mathrm{FP}^{\mathrm{PH}}$ or even $\#\mathrm{P}^{\mathrm{PH}}$ being contained in $\mathrm{FP}^{\#\mathrm{P}_1{}^{\#\mathrm{P}_1}}$? We note that the containment $\mathrm{FP}^{\mathrm{PH}} \subseteq \mathrm{FP}^{\#\mathrm{P}_1{}^{\#\mathrm{P}_1}}$ appears to be unlikely, since it would imply that $\mathrm{FP}^{\mathrm{PH}} \subseteq \mathrm{FP/poly}$. In turn, the assumption $\mathrm{FP}^{\mathrm{PH}} \subseteq \mathrm{FP/poly}$ implies that the polynomial hierarchy has polynomial-size circuits and thus collapses by the result of Karp and Lipton [KL80]. In contrast, the inclusion $\mathrm{FP}_1{}^{\mathrm{PH}} \subseteq \mathrm{FP}_1\mathrm{/poly}$, which indeed does follow from Theorem 8.3.7, merely implies that all *tally* sets in PH have polynomial-size circuits, a true statement that has no unlikely consequences. Indeed, P/poly is known to contain *all* tally sets and even the Turing closure of the sparse sets.

Third, we mention that Theorem 8.3.7 nonetheless can be strengthened. Note that our proof below will make use of Toda and Ogihara's [TO92] result that $\mathrm{PH} \subseteq \oplus\mathrm{P/poly}$. Since Toda and Ogihara [TO92] also showed that $\oplus\mathrm{P}^{\mathrm{PH}}\mathrm{/poly} = \oplus\mathrm{P/poly}$, and so $\oplus\mathrm{P}^{\mathrm{PH}} \subseteq \oplus\mathrm{P/poly}$, Theorem 8.3.7 and its corollaries could be stated even with PH replaced by $\oplus\mathrm{P}^{\mathrm{PH}}$. However, we focus on the PH case, as this is a more natural and more central class.

Now, we turn to the proof of Theorem 8.3.7.

---

[2] From part 1 of Proposition 8.2.4 and from Toda's Theorem, we have:

$$\mathrm{P}^{\mathrm{PH}} = \mathrm{P}^{\mathrm{PH}[1]} \subseteq \mathrm{P}^{\mathrm{P}^{\#\mathrm{P}[1]}[1]} = \mathrm{P}^{\#\mathrm{P}[1]}.$$

The inclusions (8.3.1) and (8.3.2) now follow from Proposition 8.2.3, the equalities $\#_1 \cdot \mathrm{P}^{\mathrm{PH}} = \#\mathrm{P}_1{}^{\mathrm{PH}}$ and $\# \cdot \mathrm{P}^{\mathrm{PH}} = \#\mathrm{P}^{\mathrm{PH}}$ from parts 2 and 3 of Proposition 8.2.4, and the similar observations that $\#_1 \cdot \mathrm{P}^{\#\mathrm{P}[1]} = \#\mathrm{P}_1{}^{\#\mathrm{P}[1]}$ and $\# \cdot \mathrm{P}^{\#\mathrm{P}[1]} = \#\mathrm{P}^{\#\mathrm{P}[1]}$.

Proof of Theorem 8.3.7. Let f be any function in $\#\mathrm{P}_1^{\mathrm{PH}}$. By part 2 of Proposition 8.2.4, we have $\#\mathrm{P}_1^{\mathrm{PH}} = \#_1 \cdot \mathrm{PH}$. Thus, there exist a set $\hat{L} \in \mathrm{PH}$ and a polynomial $\hat{p}$ such that for each length n, $f(1^n) = ||\{y \in \{0,1\}^{\hat{p}(n)} \mid 1^n\#y \in \hat{L}\}||$.

Before we proceed with the proof, a technical point need be discussed. For the calculations in the final paragraph of this proof, it would be useful to have a polynomial $\hat{p}$ satisfying that for each n, $\log \hat{p}(n)$ is an integer, i.e., $\hat{p}(n)$ is a power of two. Since that cannot be assumed in general, we define a function $p : \mathbb{N} \to \mathbb{N}$ as follows. For each n, $p(n)$ is the smallest power of 2 such that $\hat{p}(n) \leq p(n)$. Since for every integer there is a power of 2 that is at most double that integer, we have $p(n) \leq 2\hat{p}(n)$; so, p still is polynomially bounded in n. Define a padded version L of $\hat{L}$ by

$$L = \{1^n\#yw \mid n \in \mathbb{N} \ \wedge \ |y| = \hat{p}(n) \ \wedge \ 1^n\#y \in \hat{L} \ \wedge \ w = 0^{p(n)-\hat{p}(n)}\}.$$

It follows that $L \in \mathrm{PH}$ and for each length n, $f(1^n) = ||\{y \in \{0,1\}^{p(n)} \mid 1^n\#y \in L\}||$.

By Toda and Ogihara's result that $\mathrm{PH} \subseteq \oplus\mathrm{P/poly}$ [TO92], there exist a set $A \in \oplus\mathrm{P}$, an advice function $h : \Sigma^* \to \Sigma^*$, and a polynomial q such that for each length m and each x of length m, it holds that $|h(1^m)| = q(m)$, and $x \in L$ if and only if $\langle x, h(1^m)\rangle \in A$. By the argument given in the proof of Theorem 8.3.6, h is computable in $\mathrm{FP}_1^{\#\mathrm{P}_1[1]}$. Let M be a machine witnessing that $A \in \oplus\mathrm{P}$, i.e., for every string z, $z \in A$ if and only if $\mathrm{acc}_M(z)$ is odd.

Toda [Tod91b] defined inductively the following sequence of polynomials: For each $j \in \mathbb{N}$, define $s_0(j) = j$, and for each $j \in \mathbb{N}$ and $i > 0$, define

$$s_i(j) = 3(s_{i-1}(j))^4 + 4(s_{i-1}(j))^3.$$

One very useful property of this sequence of polynomials is that for all $i, j \in \mathbb{N}$, it holds that $s_i(j) = c_j \cdot 2^{2^i}$ for some $c_j \in \mathbb{N}$ if j is even, and $s_i(j) = d_j \cdot 2^{2^i} - 1$ for some $d_j \in \mathbb{N}$ if j is odd; see Toda [Tod91b] for the induction proof.

We describe a polynomial-time oracle transducer T that, on input 1^n, invokes its $\#\mathrm{P}_1^{\#\mathrm{P}_1}$ function oracle g on 1^n, receives the number $g(1^n)$ written in binary, and then prints in binary the number $f(1^n)$. Formally, function g is defined by

$$g(1^n) = \sum_{y \in \{0,1\}^{p(n)}} \left(s_{\ell_n}(\mathrm{acc}_M(\langle 1^n\#y, h(1^{n+1+p(n)})\rangle))\right)^2,$$

where $\ell_n = \log p(n)$.

Intuitively, the fact that g is in $\#\mathrm{P}_1^{\#\mathrm{P}_1}$ follows from the properties of the Toda polynomials, from the closure of #P (hence of $\#\mathrm{P}_1$) under strong sum and product,[3] and from the fact that advice function h is computable in $\mathrm{FP}_1^{\#\mathrm{P}_1[1]}$.

[3]That #P is closed under strong sum and product means the following: If $f \in \#\mathrm{P}$ and q is a polynomial, then the functions $\mathit{sum}(x) = \sum_{|y| \leq q(|x|)} f(\langle x, y\rangle)$ and $\mathit{prod}(x) = \prod_{0 \leq y \leq q(|x|)} f(\langle x, y\rangle)$ both are in #P. We refer to the work of Fenner et al. [FFK94] for a proof of this claim.

More formally, to show that $g \in \#\mathrm{P}_1^{\#\mathrm{P}_1}$, we describe a tally NP oracle machine G and a $\#\mathrm{P}_1$ oracle $\hat{g}$ for G such that, for every n, the number of accepting paths of G on input 1^n with oracle $\hat{g}$ equals $g(1^n)$.

On input 1^n, G first gets the advice string $a_n = h(1^{n+1+p(n)})$ of length $q(n+1+p(n))$ via one call to some appropriate $\#\mathrm{P}_1$ oracle, say $\hat{g}$. This is possible by the argument given in the proof of Theorem 8.3.6, which shows how to construct $\hat{g}$. Then, G guesses all strings y of length $p(n)$ and for each y guessed proceeds as follows. For fixed n and y of length $p(n)$, let $j_{n,y}$ be a shorthand for $\mathrm{acc}_M(\langle 1^n \# y, a_n \rangle)$. Note that, for given n and y of length $p(n)$, $(s_{\ell_n}(j_{n,y}))^2$ is a polynomial in $j_{n,y}$ of degree $2^{2\ell_n+1}$, which is polynomial in n. Also, the coefficients of the polynomial $(s_{\ell_n}(j_{n,y}))^2$ are deterministically computable in time polynomial in n; see Toda [Tod91b]. Since $\mathrm{acc}_M \in \#\mathrm{P}$ and $\#\mathrm{P}$ is closed under strong sum and product (see Footnote 3), the function mapping $\langle 1^n \# y, a_n \rangle$ to $(s_{\ell_n}(j_{n,y}))^2$ is in $\#\mathrm{P}$. Let $\tilde{G}$ be an NP machine witnessing that this function is in $\#\mathrm{P}$. Then, G on input 1^n can for each guessed y produce exactly $(s_{\ell_n}(j_{n,y}))^2$ accepting paths by simulating $\tilde{G}$ on input $\langle 1^n \# y, a_n \rangle$. Again using the closure of $\#\mathrm{P}$ under strong sum, it follows that $g \in \#\mathrm{P}_1^{\#\mathrm{P}_1}$, as claimed.

By the above properties of the Toda polynomials, it follows that for each n and for each y of length $p(n)$, if $j_{n,y}$ is even then $s_{\ell_n}(j_{n,y}) = c_{j_{n,y}} \cdot 2^{2^{\ell_n}}$ for some $c_{j_{n,y}} \in \mathbb{N}$, and if $j_{n,y}$ is odd then $s_{\ell_n}(j_{n,y}) = d_{j_{n,y}} \cdot 2^{2^{\ell_n}} - 1$ for some $d_{j_{n,y}} \in \mathbb{N}$.

Thus, recalling that $2^{\ell_n} = p(n)$, we have

$$\begin{aligned} j_{n,y} \text{ is even} &\implies (s_{\ell_n}(j_{n,y}))^2 = ({c_{j_{n,y}}}^2 \cdot 2^{p(n)-1})2^{p(n)+1}, \text{ and} \\ j_{n,y} \text{ is odd} &\implies (s_{\ell_n}(j_{n,y}))^2 = ({d_{j_{n,y}}}^2 \cdot 2^{p(n)-1} - d_{j_{n,y}})2^{p(n)+1} + 1. \end{aligned}$$

Defining the integer-valued functions

$$\begin{aligned} \hat{c}(n,y) &= {c_{j_{n,y}}}^2 \cdot 2^{p(n)-1}, \text{ and} \\ \hat{d}(n,y) &= {d_{j_{n,y}}}^2 \cdot 2^{p(n)-1} - d_{j_{n,y}}, \end{aligned}$$

we obtain:

$$(s_{\ell_n}(j_{n,y}))^2 = \begin{cases} \hat{c}(n,y) \cdot 2^{p(n)+1} & \text{if } j_{n,y} \text{ is even} \\ \hat{d}(n,y) \cdot 2^{p(n)+1} + 1 & \text{if } j_{n,y} \text{ is odd,} \end{cases}$$

that is, the value of $(s_{\ell_n}(j_{n,y}))^2$ is a multiple of either $2^{p(n)+1}$ or $2^{p(n)+1}+1$, depending on the parity of $j_{n,y}$. Since $f(1^n) \leq 2^{p(n)}$ and since $j_{n,y}$ is odd if and only if $1^n \# y \in L$, the rightmost $p(n)+1$ bits of the binary representation of $g(1^n)$ represent the value of $f(1^n)$. Hence, after the value $g(1^n)$ has been returned by the oracle, T can output $f(1^n)$ by printing the $p(n)+1$ rightmost bits of $g(1^n)$. This completes the proof. ∎

Since $\#\mathrm{P}_1 \subseteq \mathrm{FP}$ implies $\mathrm{FP}^{\#\mathrm{P}_1{}^{\#\mathrm{P}_1}} \subseteq \mathrm{FP}$, we have from Theorem 8.3.7 the following corollary.

Corollary 8.3.8 $\#\mathrm{P}_1 \subseteq \mathrm{FP}$ *if and only if* $\#\mathrm{P}_1{}^{\mathrm{PH}} \subseteq \mathrm{FP}$*, and in particular,* $\#\mathrm{P}_1 \subseteq \mathrm{FP}$ *if and only if* $\mathrm{span}\mathrm{P}_1 \subseteq \mathrm{FP}$.

Corollary 8.3.8 together with the equivalences of Theorems 8.3.4 and 8.3.5 gives the following.

Corollary 8.3.9 *Every* P *set has an easy census function if and only if every set in* PH *has an easy census function.*

Köbler et al. [KST89] proved that $\mathrm{span}\mathrm{P} = \#\mathrm{P}$ if and only if NP = UP. Their proof also establishes the analogous result for tally sets:

Lemma 8.3.10 (implicit in [KST89]) $\mathrm{span}\mathrm{P}_1 = \#\mathrm{P}_1$ *if and only if every tally* NP *set is in* UP.

Using Lemma 8.3.10, we now show that $\mathrm{span}\mathrm{P}_1$ and $\#\mathrm{P}_1$ are different classes unless NE = UE, or unless every sparse set in NP is low for SPP. A set S is said to be $\mathcal{C}$*-low* for some class $\mathcal{C}$ if $\mathcal{C}^S = \mathcal{C}$; see, e.g., the papers [Sch83, KS85, Sch87, KSTT92] for a number of important lowness results. In particular, it is known that every sparse NP set is low for P^{NP} [KS85] and for PP [KSTT92], but it is not known whether all sparse NP sets are low for SPP. Torán's result that in some relativized world there exists some sparse NP set that is not contained in $\oplus\mathrm{P}$ [Tor88], and thus not in SPP, may be taken as evidence that not all sparse NP sets are SPP-low. Since Corollary 8.3.11 relativizes, $\mathrm{span}\mathrm{P}_1 \neq \#\mathrm{P}_1$ holds relative to the same oracle.

Corollary 8.3.11 *If* $\mathrm{span}\mathrm{P}_1 = \#\mathrm{P}_1$*, then the following two statements are true.*

1. NE = UE.
2. *Every sparse* NP *set is low for* SPP.

Proof. The first part follows from a standard upward translation argument (as mentioned in the proof of Theorem 8.3.4).

For the second part, assume $\mathrm{span}\mathrm{P}_1 = \#\mathrm{P}_1$, and let S be any sparse set in NP. By the result of Hartmanis, Immerman, and Sewelson [Har83b, HIS85], S polynomial-time truth-table reduces to some tally NP set T. By Lemma 8.3.10, our assumption implies that $T \in$ UP, and thus $T \in$ SPP. Since $\mathrm{P}^{\mathrm{SPP}} = \mathrm{SPP}$, we have $S \in$ SPP. The result now follows from the self-lowness of SPP [FFK94], i.e., from the equality $\mathrm{SPP}^{\mathrm{SPP}} = \mathrm{SPP}$. ∎

8.4 Enumerative Approximation of Census Functions

Cai and Hemaspaandra [CH89] introduced the notion of enumerative counting as a way of approximating the value of a #P function deterministically in polynomial time.

Definition 8.4.1 [CH89] *Let $f : \Sigma^* \to \Sigma^*$ and $g : \mathbb{N} \to \mathbb{N}$ be two functions. A Turing transducer E is a* $g(n)$-enumerator *of f if for all $n \in \mathbb{N}$ and $x \in \Sigma^n$,*

1. *E on input x prints a list $\mathcal{L}_x$ with at most $g(n)$ elements, and*
2. *$f(x)$ is a member of list $\mathcal{L}_x$.*

A function f is $g(n)$-enumerable *in time $t(n)$ if there exists a $g(n)$-enumerator of f that runs in time $t(n)$.*

A set is $g(n)$-enumeratively rankable *in time $t(n)$ if its ranking function is $g(n)$-enumerable in time $t(n)$.*

Recall from the introduction Hemaspaandra and Rudich's [HR90] result that every P set is k-enumeratively rankable for some fixed k (and indeed, even $\mathcal{O}(n^{1/2-\epsilon})$-enumeratively rankable for some $\epsilon > 0$) in polynomial time if and only if #P = FP. They conclude that it is no more likely that one can enumeratively rank all sets in P than that one can exactly compute their ranking functions in polynomial time. We similarly characterize the question of whether every P set has a census function that is n^α-enumerable in time n^β for fixed constants α and β. By the argument given in the proof of Theorem 8.3.4, this question is equivalent to asking whether every $\#\mathrm{P}_1$ function is n^α-enumerable in time n^β. We show that this assumption implies $\#\mathrm{P}_1 \subseteq$ FP, and we conclude that it is no more likely that one can n^α-enumerate the census function of every P set in time n^β than that one can precisely compute its census function in polynomial time. It would be interesting to know if this result can be improved to hold for polynomial time instead of time t for some fixed polynomial $t(n) = n^\beta$.

Theorem 8.4.2 *Let $\alpha, \beta > 0$ be constants. If every $\#\mathrm{P}_1$ function is n^α-enumerable in time n^β, then $\#\mathrm{P}_1 \subseteq$ FP.*

Proof. Cai and Hemaspaandra [CH91] show that for any fixed k, if #SAT (the function mapping any boolean formula f to the number of satisfying assignments of f) is n^k-enumerable, then #P $\subseteq$ FP. In order to prove this, they develop the following protocol for computing the permanent of an $m \times m$ matrix A,[4] given as parameters (the encoding of)

[4]Denoting the (i, j) entry of an $m \times m$ integer matrix A by a_{ij}, the *permanent* of A is defined to be $perm(A) = \sum_\sigma \prod_{i=1}^m a_{i\sigma(i)}$ summed over all permutations σ on $\{1, 2, \ldots, m\}$. Valiant [Val79a] showed that computing the permanent is #P-complete, i.e., *perm* $\in$ #P and #P $\subseteq \mathrm{FP}^{perm}$, where *perm* is used as a function oracle.

a polynomial-time transducer E (the enumerator for #SAT), and a prime number p: Set $A_0 = A$ to the input matrix and repeat the following steps for $i = 1, \ldots, m-1$:

1. Construct from A_{i-1} an $(m-i) \times (m-i)$ matrix $B_i(X)$ over an indeterminate X, defined by

$$B_i(X) = \sum_{k=1}^{m-i} e_k(X) a_{1k} A_{i-1}^{(1,k)},$$

   where $e_k(X)$ is a degree $(m-i)$ polynomial in X such that $e_k(X) \equiv 1$ if $X = k$ and 0 otherwise, a_{1k} is the $(1,k)$ entry of A_{i-1}, and $A_{i-1}^{(1,k)}$ is the $(1,k)$-minor of A_{i-1}. Each matrix is viewed as a matrix over $\mathbb{Z}/p\mathbb{Z}[X]$, that is, the matrix entries are polynomials in X whose integer coefficients are reduced modulo p. Then the following conditions hold.

   - Each entry of $B_i(X)$ is a degree $(m-i)$ polynomial in X with coefficients in $\{0, \ldots, p-1\}$, so *perm*$(B_i(X))$ is a degree $(m-i)^2$ polynomial in X.
   - $\sum_{k=1}^{m-i}$ *perm*$(B_i(k))$ = *perm*(A_{i-1}).

2. Encode $B_i(X)$ into a binary string specifying in binary p, m, and the coefficients of $B_i(X)$. There is some fixed constant $c > 0$ such that the encoding length is at most $c(m-i)^3 \log p$. Define $Q_i(X) =$ *perm*$(B_i(X))$. Then, Q_i is a polynomial of degree at most $(m-i)^2$, whose coefficients are each length-bounded by a fixed polynomial in p and m. Thus, there is a #P function G that maps $B_i(X)$ to a number from which the coefficients of Q_i can be decoded in polynomial time.

3. Use E as an enumerator for G to obtain candidates $g_1, \ldots, g_t$. These are all degree $(m-i)^2$ polynomials that are pairwise distinct. Since two distinct degree $(m-i)^2$ polynomials can agree at no more than $(m-i)^2 - 1$ points, there are fewer than $t^2(m-i)^2 \leq t^2 m^2 - 1$ points X at which any two candidate polynomials agree. Thus, if $p \geq t^2 m^2$, then there is an $r \in \{0, \ldots, p-1\}$ such that $g_j(r) \neq g_k(r)$ for all $j \neq k$. Take the smallest such r and set A_i to $B_i(r)$ with the entries reduced modulo p. Now, *perm*(A_i) modulo p specifies which g_j is correct, so we can recover *perm*(A_{i-1}) modulo p in polynomial time.

At the end of this loop, A_m is a 1×1 matrix, so its permanent is easy to compute. Now working backwards again, we can recover *perm*(A) modulo p. If we do this for polynomially (in the encoding length of A) many distinct primes, then by the Chinese Remainder Theorem, we can recover the exact value of *perm*(A).

Valiant [Val79a] showed that the permanent of matrices whose entries are from the set $\{-1, 0, 1, 2\}$ is complete for #P. Analogously, we can show that there exists an infinite sequence of matrices $[M_1, M_2, \dots]$ such that (i) the mapping $1^n \to \mathit{perm}(M_n)$ is complete for $\#\mathrm{P}_1$, (ii) the mapping $1^n \to M_n$ is polynomial-time computable, and (iii) for every n, M_n is an $n \times n$ matrix whose entries are from $\{-1, 0, 1, 2\}$. Because of (iii), $\mathit{perm}(M_n) \leq 2^{2n}$ for all n. So, by the Chinese Remainder Theorem, for every n, the exact value of $\mathit{perm}(M_n)$ can be computed from $\mathit{perm}(M_n)$ modulo p for $2n$ arbitrary distinct primes p. Define polynomials q and s by $q(n) = \langle n, n, n, 2n\rangle$ and $s(n) = q(n)^{2\alpha}n^2$. Define the function f from the tally strings to the set of natural numbers as follows.

- If $m = \langle H, n, i, j\rangle$ for some H, $i \leq n$, and $j \leq 2n$, then $f(1^m)$ is $G(B_i(X))$, where the function G and the matrix $B_i(X)$ are defined as in the above protocol, except that now we simulate the protocol subject to the following constraints:
  - The jth smallest prime $> s(n)$ is used in place of p.
  - M_n is used in place of the input matrix A_0.
  - H is viewed as (the encoding of) a Turing transducer and is used in place of the enumerator E. Here, for each k with $1 \leq k \leq i-1$, the input given to H in the kth round of the protocol is $\langle H, n, k, j\rangle$, not the matrix A_k. Also, H is supposed to run in $q(n)^\beta$ steps and to generate at most $q(n)^\alpha$ candidates in each round. If H does not halt in $q(n)^\beta$ steps or generates more than $q(n)^\alpha$ candidates at any point of the simulation, then the simulation is immediately aborted and the value $f(1^m)$ is set to 0.
- If m is not of the above form, $f(1^m)$ is 0.

This function f is in $\#\mathrm{P}_1$. First, there are only $i \leq m$ rounds to be simulated and each round requires m^α steps for candidate generation and some polynomial (in n) number of steps for other computations. Second, by the Prime Number Theorem, the first $2n$ smallest primes $> n$ are in $\mathcal{O}(n)$. (Remember that m is the *length* of the input and m is bounded by some polynomial in n; so, in time polynomial in n one can find these primes using simple methods such as the Sieve of Erastosthenes.) Since j and $s(n)$ are polynomially bounded, finding the jth smallest prime $> s(n)$ requires only a polynomial number of steps.

Now, by our assumption, there is an m^α-enumerator $\widehat{E}$ for f that runs in time m^β. Since the number of candidates that $\widehat{E}$ generates is at most m^α and the dimension of the matrix M_n is n, we have a prime $> m^{2\alpha}n^2$. This implies that with $\widehat{E}$ as the enumerator, for every $n \geq \widehat{E}$, every $j, 1 \leq j \leq 2n$, and every $i, 1 \leq i \leq n$, we successfully find an r for distinguishing the candidates. So, with $\widehat{E}$ as the enumerator, for all $n \geq \widehat{E}$, $\mathit{perm}(M_n)$ is polynomial-time computable. Hence $\#\mathrm{P}_1 \subseteq \mathrm{FP}$. ∎

8.5 Oracle Results

In this section, we provide a number of relativized results on the existence or non-existence of P sets simultaneously satisfying pairs of conditions chosen among the properties (i), (ii), and (iii) from Section 8.3. For instance, Theorem 8.5.1 and its Corollary 8.5.2 below exhibit a relativized world in which every P set has an easy census function (Property (ii)), yet there exists some set in P that is not rankable (Property (i)).

Theorem 8.5.1 *There exists an oracle D such that* $\#\mathrm{P}_1^D \subseteq \mathrm{FP}^D \neq \#\mathrm{P}^D$.

From the relativized versions of Theorem 8.3.4 and of Hemaspaandra and Rudich's result in [HR90] that every P set is rankable if and only if $\mathrm{P}^{\#\mathrm{P}} = \mathrm{P}$ (which is equivalent with $\mathrm{FP} = \#\mathrm{P}$, and this equivalence itself also relativizes), we immediately obtain the following corollary.

Corollary 8.5.2 *There exists an oracle D such that all sets in* P^D *have a census function computable in* FP^D*, yet there exists some set in* P^D *that is not rankable by any function in* FP^D.

Proof of Theorem 8.5.1. Balcázar et al. [BBS86] and Long and Selman [LS86] proved that the polynomial hierarchy does not collapse if and only if it does not collapse relative to every sparse oracle. Since their proof relativizes (i.e., it applies to the relativized polynomial hierarchy as well), we have the following claim:

Claim 8.5.3 [BBS86, LS86] *For every set B,* PH^B *does not collapse if and only if for every sparse oracle S,* $(\mathrm{PH}^B)^S$ *does not collapse.*

Note that $(\mathrm{PH}^B)^S = \mathrm{PH}^{B\oplus S}$. Fix an oracle A such that PH^A does not collapse (such oracles were constructed by Yao [Yao85], Hastad [Has89], and Ko [Ko89] who built on the work of Furst et al. [FSS84]). Then, by Claim 8.5.3 above, for every sparse set S, $\mathrm{PH}^{A\oplus S}$ does not collapse. So, in particular, $\mathrm{P}^{A\oplus S} \neq \mathrm{NP}^{A\oplus S}$ for every sparse set S. Since for every oracle B, $\#\mathrm{P}^B = \mathrm{FP}^B$ implies $\mathrm{NP}^B = \mathrm{P}^B$, we have that $\#\mathrm{P}^{A\oplus S} \neq \mathrm{FP}^{A\oplus S}$ for every sparse set S.

So it remains to prove that there exists a sparse set T such that $\#\mathrm{P}_1^{A\oplus T} \subseteq \mathrm{FP}^{A\oplus T}$. Then, setting $D = A \oplus T$ completes the proof.

Recall that our pairing function $\langle \cdot, \cdot, \cdot \rangle$ is non-decreasing in each argument, polynomial-time computable and invertible, and is one-to-one and onto. Let $N_1^{(\cdot)}, N_2^{(\cdot)}, \ldots$ be a standard enumeration of all tally NP oracle machines. For each $i \geq 1$, let p_i be the polynomial time bound of $N_i^{(\cdot)}$. Then, the function $f^{(\cdot)}$ defined by

$$f^{(\cdot)}(1^{\langle i,n,j\rangle}) \quad = \quad \begin{cases} \mathrm{acc}_{N_i^{(\cdot)}}(1^n) & \text{if } p_i(n) < j \\ 0 & \text{otherwise} \end{cases}$$

is a canonical function complete for the class $\#\mathrm{P}_1^{(\cdot)}$.[5] In particular, for every fixed set S, $f^{(A\oplus S)}$ is complete for $\#\mathrm{P}_1^{A\oplus S}$.

The oracle set T is defined in such a way that, for any given $m = \langle i, n, j\rangle$ in unary, some polynomial-time oracle transducer can retrieve the value of $f^{(A\oplus T)}(1^m)$ from its oracle $A \oplus T$ by asking at most m queries. More formally, we construct T in stages such that for each $m = \langle i, n, j\rangle$:

$$1^k 0^{m-k} \# b \in T \iff 1 \le k \le |f^{(A\oplus T)}(1^m)| \text{ and the } k\text{th bit of } f^{(A\oplus T)}(1^m) \text{ is } b.$$

By the above definition, $|f^{(A\oplus T)}(1^m)| < m$ and coding information into the oracle is unnecessary when $N_i^{A\oplus T}(1^n)$ queries strings of length $\ge m$. So there is no interference between the stages of the construction of T. It is easy to see that T is a sparse set satisfying $\#\mathrm{P}_1^{A\oplus T} \subseteq \mathrm{FP}^{A\oplus T}$. ∎

Now we construct an oracle relative to which there exists some scalable set in P whose census function is not easy to compute.

Theorem 8.5.4 *There exists an oracle A such that there exists an A-scalable set B whose census function is not in* FP^A.

Proof. We will construct A and B in such a way that B is P^A-isomorphic to the set $R = \{0x \mid x \in \Sigma^*\}$, which is rankable in FP (and thus in FP^A). For each $n \ge 1$, we have $\mathit{census}_R(1^n) = 2^{n-1}$. So census_R is easy to compute, but we want B to have a hard census function. In light of part 2 of Proposition 8.3.1, we thus need the isomorphism, f, between B and R to be non-length-preserving. In particular, we will define f so as to satisfy $|f(x)| \le |x| + 1$ and $|f^{-1}(y)| \le |y|$ for all $x, y \in \Sigma^*$. When f is defined, we let B be the set $f^{-1}(R)$. To have f and its inverse computable in FP^A, we encode f and f^{-1} into $A = A_f \oplus A_{f^{-1}}$ as follows. For all $x \in \Sigma^*$, $i \ge 1$, and $b \in \{0, 1\}$, we ensure that

$$\langle x, i, b\rangle \in A_{f^*} \iff \text{the } i\text{th bit of } f^*(x) \text{ is } b, \tag{8.5.3}$$

where f^* stands for either f or f^{-1}. At the same time we diagonalize against FP^A so as to ensure $\mathit{census}_B \notin \mathrm{FP}^A$.

Let $T_1^{(\cdot)}, T_2^{(\cdot)}, \ldots$ be a standard enumeration of all deterministic polynomial-time oracle transducers, and let $p_1, p_2, \ldots$ be a sequence of strictly increasing polynomials such that p_i bounds the running time of T_i (independent of the oracle used). By (8.5.3) above, implicit in the definition of f and f^{-1} is the definition of A, so it suffices to construct the isomorphism. The construction of f and f^{-1} is in stages. By the end of stage i, f will have been defined for all strings of length up to $r(i)$, where r will be determined below. Initially, we start with $r(0) = 0$, and we define $f(\epsilon) = \epsilon$. Stage $i > 0$ of the construction is as follows.

[5] See [Val79b] for *natural* $\#\mathrm{P}_1$-complete functions.

Stage i: Choose n_i to be the smallest integer such that $n_i > r(i-1)$ and $p_i(n_i) < 2^{n_i-2}$. Let A' be the subset of A that has been decided by now. We want to define f so that, eventually, $T_i^A(1^{n_i}) \neq \mathit{census}_B(1^{n_i})$. Simulate $T_i^{A'}$ on input 1^{n_i}. Whenever in this simulation a string of the form $0\langle x,i,b\rangle$ whose membership in A has not yet been decided is queried, we add this string to A' and set the ith bit of $f(x)$ to b unless we have already put $0\langle x,i,1-b\rangle$ into A (and thus have set this bit to $1-b$), or unless $i > |x|+1$. The same comment applies to query strings $1\langle y,j,b\rangle$ whose membership in A has not been decided yet and which may fix the jth bit of $f^{-1}(y)$. If we added the queried string to A', we continue the simulation in the "yes" state; otherwise, in the "no" state. In this way, the simulation of $T_i^{A'}(1^{n_i})$ may determine f (and f^{-1}) on at most $p_i(n_i) < 2^{n_i-2}$ bits of the strings of length n_i. Thus, for no $m \geq n_i$ is f^{-1} determined on all strings of length m in R or $\overline{R}$. Once the value $T_i^{A'}(1^{n_i})$ is computed, there is room to decide $f(x)$ and $f^{-1}(y)$ for all strings x and y of lengths between $r(i-1)$ and $p_i(n_i)$ so that f is an isomorphism mapping to $\bigcup_{\ell=r(i-1)}^{p_i(n_i)} R^{=\ell}$ and such that $census_B(1^{n_i}) \neq T_i^{A'}(1^{n_i})$, without changing the output value of $T_i^{A'}(1^{n_i})$. Finally, define $r(i) = p_i(n_i)$. ∎

Next, we provide an oracle relative to which there exists some set in P that is neither scalable nor has an easy census function.

Theorem 8.5.5 *There exists an oracle D such that $D \in \mathrm{P}^D$ is not D-scalable and its census function is not in* FP^D.

Proof. It is known from the work of Goldsmith and Homer [GH96] that any sparse set is scalable if and only if it is rankable, and this holds if and only if it is P-printable. D will be sparse, with at most 2 strings at each length. We assume that $(T_i^{(\cdot)})_{i\geq 1}$ enumerates $\mathrm{FP}^{(\cdot)}$, and that $T_i^{(\cdot)}$ runs in time n^i. A simple diagonalization guarantees that no P^D function computes the census of D. Note that this guarantees that no P^D function computes the rank of 1^n in D for all n, since $\mathit{census}_D(1^n) = \mathit{rank}_D(1^n) - \mathit{rank}_D(1^{n-1})$ would then be in P^D.

At stage i we guarantee that $T_i^D(1^n)$ does not compute the census function of D, where n is chosen large enough that $n^i < 2^n$. Compute $T_i^D(1^n)$, restraining any oracle strings of length $\geq n$ that it queries. By our choice of n, this does not decide $D^{=m}$ for any $m \geq n$, so we can then put in the appropriate number of strings of length n for the diagonalization. ∎

Finally, we show that relative to an oracle, there exists some non-scalable set in P having an easy census function.

Theorem 8.5.6 *There exists an oracle A such that $A \in \mathrm{P}^A$ is not A-scalable and its census function is in* FP^A.

Proof. We construct the oracle A so that A has one string of each length. For those lengths for which nothing else is decided, we put in 1^n. Otherwise, we do the following.

To make the oracle A non-A-scalable, we actually make it non-P^A-printable. At stage i, choose an appropriate length n, and then compute $T_i^A(1^n)$. Whenever it queries a string of length $\geq n$, restrain the string from the oracle. If it does anything except print out $A^{\leq n}$, then put in the first unrestrained string of each length. If it correctly prints A up to length n, then choose an x of each relevant length to include that neither is restrained nor printed. ∎

Index

Bibliography

[Adl78] L. Adleman. Two theorems on random polynomial time. In *Proceedings of the 19th IEEE Symposium on Foundations of Computer Science*, pages 75–83, 1978.

[Adl79] L. Adleman. Time, space, and randomness. Technical Report MIT/LCS/TM-131, MIT, Cambridge, MA, April 1979.

[AH77a] K. Appel and W. Haken. Every planar map is 4-colorable – 1: Discharging. *Illinois J. Math*, 21:429–490, 1977.

[AH77b] K. Appel and W. Haken. Every planar map is 4-colorable – 2: Reducibility. *Illinois J. Math*, 21:491–567, 1977.

[All85] E. Allender. Invertible functions, 1985. PhD thesis, Georgia Institute of Technology.

[All86] E. Allender. The complexity of sparse sets in P. In *Proceedings of the 1st Structure in Complexity Theory Conference*, pages 1–11. Springer-Verlag *Lecture Notes in Computer Science #223*, June 1986.

[All90] E. Allender. Oracles versus proof techniques that do not relativize. In *Proceedings of the 1990 SIGAL International Symposium on Algorithms*, pages 39–52. Springer-Verlag *Lecture Notes in Computer Science #450*, August 1990.

[All91] E. Allender. Limitations of the upward separation technique. *Mathematical Systems Theory*, 24(1):53–67, 1991.

[All92] E. Allender. Applications of time-bounded Kolmogorov complexity in complexity theory. In O. Watanabe, editor, *Kolmogorov Complexity and Computational Complexity*, EATCS Monographs on Theoretical Computer Science, pages 4–22. Springer-Verlag, 1992.

[AR88] E. Allender and R. Rubinstein. P-printable sets. *SIAM Journal on Computing*, 17(6):1193–1202, 1988.

[Bal85] J. Balcázar. Simplicity, relativizations and nondeterminism. *SIAM Journal on Computing*, 14(1):148–157, 1985.

[BB86] J. Balcázar and R. Book. Sets with small generalized Kolmogorov complexity. *Acta Informatica*, 23(6):679–688, 1986.

[BBS86] J. Balcázar, R. Book, and U. Schöning. The polynomial-time hierarchy and sparse oracles. *Journal of the ACM*, 33(3):603–617, 1986.

[BC93] D. Bovet and P. Crescenzi. *Introduction to the Theory of Complexity*. Prentice Hall, 1993.

[BCO93] R. Beigel, R. Chang, and M. Ogiwara. A relationship between difference hierarchies and relativized polynomial hierarchies. *Mathematical Systems Theory*, 26(3):293–310, 1993.

[BCS92] D. Bovet, P. Crescenzi, and R. Silvestri. A uniform approach to define complexity classes. *Theoretical Computer Science*, 104(2):263–283, 1992.

[BD76] A. Borodin and A. Demers. Some comments on functional self-reducibility and the NP hierarchy. Technical Report TR 76-284, Cornell Department of Computer Science, Ithaca, NY, July 1976.

[BDG88] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*. EATCS Monographs in Theoretical Computer Science. Springer-Verlag, 1988.

[BDG90] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity II*. EATCS Monographs in Theoretical Computer Science. Springer-Verlag, 1990.

[Bei89] R. Beigel. On the relativized power of additional accepting paths. In *Proceedings of the 4th Structure in Complexity Theory Conference*, pages 216–224. IEEE Computer Society Press, June 1989.

[Bei91] R. Beigel. Relativized counting classes: Relations among thresholds, parity, and mods. *Journal of Computer and System Sciences*, 42(1):76–96, 1991.

[Bei94] R. Beigel. Perceptrons, PP, and the polynomial hierarchy. *Computational Complexity*, 4(4):339–349, 1994.

[Ber97] A. Berthiaume. Quantum computation. In L. Hemaspaandra and A. Selman, editors, *Complexity Theory Retrospective II*, pages 23–51. Springer-Verlag, 1997.

[BG81] C. Bennett and J. Gill. Relative to a random oracle A, $P^A \neq NP^A \neq coNP^A$ with probability 1. *SIAM Journal on Computing*, 10(1):96–113, 1981.

[BG92] R. Beigel and J. Gill. Counting classes: Thresholds, parity, mods, and fewness. *Theoretical Computer Science*, 103(1):3–23, 1992.

[BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the P=?NP question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[BH77] L. Berman and J. Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM Journal on Computing*, 6(2):305–322, 1977.

[BHR99] B. Borchert, L. Hemaspaandra, and J. Rothe. Restrictive acceptance suffices for equivalence problems. In *Proceedings of the 12th International Symposium on Fundamentals of Computation Theory*. Springer Verlag *Lecture Notes in Computer Science #1684*, August/September 1999. To appear.

[BJY90] D. Bruschi, D. Joseph, and P. Young. Strong separations for the boolean hierarchy over RP. *International Journal of Foundations of Computer Science*, 1(3):201–218, 1990.

[Bla58] D. Black. *The Theory of Committees and Elections*. Cambridge University Press, 1958.

[Boo74] R. Book. Tally languages and complexity classes. *Information and Control*, 26(2):186–193, 1974.

[BR88] J. Balcázar and D. Russo. Immunity and simplicity in relativizations of probabilistic complexity classes. *R.A.I.R.O. Theoretical Informatics and Applications*, 22(2):227–244, 1988.

[Bru92] D. Bruschi. Strong separations of the polynomial hierarchy with oracles: Constructive separations by immune and simple sets. *Theoretical Computer Science*, 102(2):215–252, 1992.

[BS97] B. Borchert and F. Stephan. Looking for an analogue of Rice's Theorem in circuit complexity theory. In *Proceedings on the 1997 Kurt Gödel Colloquium*, pages 114–127. Springer-Verlag *Lecture Notes in Computer Science #1289*, 1997.

[BTT89a] J. Bartholdi III, C. Tovey, and M. Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6:227–241, 1989.

[BTT89b] J. Bartholdi III, C. Tovey, and M. Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6:157–165, 1989.

[BTT92] J. Bartholdi III, C. Tovey, and M. Trick. How hard is it to control an election? *Mathematical Comput. Modelling*, 16(8/9):27–40, 1992.

[BTY97] H. Bodlaender, D. Thilikos, and K. Yamazaki. It is hard to know when greedy is good for finding independent sets. *Information Processing Letters*, 61:101–106, 1997.

[BU] C. Berg and S. Ulfberg. A lower bound for perceptrons and an oracle separation of the PP^{PH} hierarchy. *Journal of Computer and System Sciences*. To appear. A preliminary version appeared in the *Proceedings of the 12th Annual IEEE Conference on Computational Complexity*, pages 165–172. IEEE Computer Society Press, 1997.

[CGH+88] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The boolean hierarchy I: Structural properties. *SIAM Journal on Computing*, 17(6):1232–1252, 1988.

[CGH+89] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The boolean hierarchy II: Applications. *SIAM Journal on Computing*, 18(1):95–111, 1989.

[CH89] J. Cai and L. Hemachandra. Enumerative counting is hard. *Information and Computation*, 82(1):34–44, 1989.

[CH90] J. Cai and L. Hemachandra. On the power of parity polynomial time. *Mathematical Systems Theory*, 23(2):95–106, 1990.

[CH91] J. Cai and L. Hemachandra. A note on enumerative counting. *Information Processing Letters*, 38(4):215–219, 1991.

[Cha66] G. Chaitin. On the length of programs for computing finite binary sequences. *Journal of the ACM*, 13:547–569, 1966.

[CHV93] J. Cai, L. Hemachandra, and J. Vyskoč. Promises and fault-tolerant database access. In K. Ambos-Spies, S. Homer, and U. Schöning, editors, *Complexity Theory*, pages 101–146. Cambridge University Press, 1993.

[Cob64] A. Cobham. The intrinsic computational difficulty of functions. In *Proceedings of the 1964 International Congress for Logic Methodology and Philosophy of Science*, pages 24–30. North Holland, 1964.

[Coo71] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[CS93] P. Crescenzi and R. Silvestri. Sperner's lemma and robust machines. In *Proceedings of the 8th Structure in Complexity Theory Conference*, pages 194–199. IEEE Computer Society Press, 1993.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[DH79] W. Diffie and M. Hellman. Privacy and authentication: An introduction to cryptography. *Proceedings of the IEEE,* 67(3):397–427, 1979.

[Dod76] C. Dodgson. A method of taking votes on more than two issues. Pamphlet printed by the Clarendon Press, Oxford, and headed "not yet published" (see the discussions in [MU95, Bla58], both of which reprint this paper), 1876.

[Edm65] J. Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[EHTY92] D. Eppstein, L. Hemachandra, J. Tisdall, and B. Yener. Simultaneous strong separations of probabilistic and unambiguous complexity classes. *Mathematical Systems Theory*, 25(1):23–36, 1992.

[ESY84] S. Even, A. Selman, and Y. Yacobi. The complexity of promise problems with applications to public-key cryptography. *Information and Control*, 61(2):159–173, 1984.

[EY80] S. Even and Y. Yacobi. Cryptocomplexity and NP-completeness. In *Proceedings of the 7th International Colloquium on Automata, Languages, and Programming*, pages 195–207. Springer-Verlag *Lecture Notes in Computer Science*, 1980.

[FFK94] S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. *Journal of Computer and System Sciences*, 48(1):116–148, 1994.

[FFNR96] S. Fenner, L. Fortnow, A. Naik, and J. Rogers. On inverting onto functions. In *Proceedings of the 11th Annual IEEE Conference on Computational Complexity*, pages 213–222. IEEE Computer Society Press, May 1996.

[FK92] M. Fellows and N. Koblitz. Self-witnessing polynomial-time complexity and prime factorization. In *Proceedings of the 7th Structure in Complexity Theory Conference*, pages 107–110. IEEE Computer Society Press, June 1992.

[FK96] U. Feige and J. Kilian. Zero knowledge and the chromatic number. In *Proceedings of the 11th Annual IEEE Conference on Computational Complexity*, pages 278–287. IEEE Computer Society Press, 1996.

[For94] L. Fortnow. The role of relativization in complexity theory. *Bulletin of the EATCS*, 52:229–244, 1994.

[For97a] L. Fortnow, 1997. Personal Communication.

[For97b] L. Fortnow. Counting complexity. In L. Hemaspaandra and A. Selman, editors, *Complexity Theory Retrospective II*, pages 81–107. Springer-Verlag, 1997.

[FR91] L. Fortnow and N. Reingold. PP is closed under truth-table reductions. In *Proceedings of the 6th Structure in Complexity Theory Conference*, pages 13–15. IEEE Computer Society Press, June/July 1991.

[FR94] L. Fortnow and J. Rogers. Separability and one-way functions. In *Proceedings of the 5th International Symposium on Algorithms and Computation*, pages 396–404. Springer-Verlag *Lecture Notes in Computer Science #834*, August 1994.

[FSS84] M. Furst, J. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

[GH96] J. Goldsmith and S. Homer. Scalability and the isomorphism problem. *Information Processing Letters*, 57(3):137–143, 1996.

[GHJY91] J. Goldsmith, L. Hemachandra, D. Joseph, and P. Young. Near-testable sets. *SIAM Journal on Computing*, 20(3):506–523, 1991.

[GHK92] J. Goldsmith, L. Hemachandra, and K. Kunen. Polynomial-time compression. *Computational Complexity*, 2(1), 1992.

[Gil77] J. Gill. Computational complexity of probabilistic Turing machines. *SIAM Journal on Computing*, 6(4):675–695, 1977.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[GJS76] M. Garey, D. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.

[GJY87] J. Goldsmith, D. Joseph, and P. Young. Self-reducible, P-selective, near-testable, and P-cheatable sets: The effect of internal structure on the complexity of a set. In *Proceedings of the 2nd Structure in Complexity Theory Conference*, pages 50–59, 1987.

[GLS84] M. Grötschel, L. Lovász, and A. Schrijver. Polynomial algorithms for perfect graphs. In *Perfect Graphs*, volume 21 of *Annals of Discrete Mathematics*, pages 325–356. North-Holland Publishing Co., Amsterdam, 1984.

[GNW90] T. Gundermann, N. Nasser, and G. Wechsung. A survey on counting classes. In *Proceedings of the 5th Structure in Complexity Theory Conference*, pages 140–153. IEEE Computer Society Press, July 1990.

[Gol89] J. Goldsmith. *Polynomial Isomorphisms and Near-Testable Sets*. PhD thesis, University of Wisconsin–Madison, Madison, WI, January 1989. Available as Technical Report 816.

[GOR] J. Goldsmith, M. Ogihara, and J. Rothe. Tally NP sets and easy census functions. *Information and Computation*. To appear.

[GOR98] J. Goldsmith, M. Ogihara, and J. Rothe. Tally NP sets and easy census functions. In *Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science*, pages 483–492. Springer-Verlag *Lecture Notes in Computer Science #1450*, August 1998.

[GP86] L. Goldschlager and I. Parberry. On the construction of parallel computers from various bases of boolean functions. *Theoretical Computer Science*, 43(1):43–58, 1986.

[Grä94] E. Grädel. Definability on finite structures and the existence of one-way functions. *Methods of Logic in Computer Science*, 1:299–314, 1994.

[Gre91] F. Green. An oracle separating $\oplus$P from $\mathrm{PP}^{\mathrm{PH}}$. *Information Processing Letters*, 37(3):149–153, 1991.

[GS88] J. Grollmann and A. Selman. Complexity measures for public-key cryptosystems. *SIAM Journal on Computing*, 17(2):309–335, 1988.

[GS91] A. Goldberg and M. Sipser. Compression and ranking. *SIAM Journal on Computing*, 20(3):524–536, 1991.

[GT91] F. Green and J. Torán. Kolmogorov complexity of #P functions. Manuscript, 1991.

[GW87] T. Gundermann and G. Wechsung. Counting classes with finite acceptance types. *Computers and Artificial Intelligence*, 6(5):395–409, 1987.

[Har83a] J. Hartmanis. Generalized Kolmogorov complexity and the structure of feasible computations. In *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, pages 439–445. IEEE Computer Society Press, 1983.

[Har83b] J. Hartmanis. On sparse sets in NP−P. *Information Processing Letters*, 16(2):55–60, 1983.

[Har85] J. Hartmanis. Solvable problems with conflicting relativizations. *Bulletin of the EATCS*, 27:40–49, 1985.

[Has89] J. Hastad. Almost optimal lower bounds for small depth circuits. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 143–170. JAI Press, Greenwich, 1989.

[Hem87] L. Hemachandra. *Counting in Structural Complexity Theory*. PhD thesis, Cornell University, Ithaca, NY, May 1987. Available as Cornell Department of Computer Science Technical Report TR87-840.

[Hem89] L. Hemachandra. The strong exponential hierarchy collapses. *Journal of Computer and System Sciences*, 39(3):299–322, 1989.

[Her90] U. Hertrampf. Relations among MOD-classes. *Theoretical Computer Science*, 74(3):325–328, 1990.

[HH76] J. Hartmanis and J. Hopcroft. Independence results in computer science. *SIGACT News*, 8(4):13–24, 1976.

[HH88a] J. Hartmanis and L. Hemachandra. Complexity classes without machines: On complete languages for UP. *Theoretical Computer Science*, 58:129–142, 1988.

[HH88b] J. Hartmanis and L. Hemachandra. On sparse oracles separating feasible complexity classes. *Information Processing Letters*, 28:291–295, 1988.

[HH90] J. Hartmanis and L. Hemachandra. Robust machines accept easy sets. *Theoretical Computer Science*, 74(2):217–226, 1990.

[HH91] J. Hartmanis and L. Hemachandra. One-way functions and the nonisomorphism of NP-complete sets. *Theoretical Computer Science*, 81(1):155–163, 1991.

[HHH99] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. A downward collapse within the polynomial hierarchy. *SIAM Journal on Computing*, 28(2):383–393, 1999.

[HHR97a] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. *Journal of the ACM*, 44(6):806–825, 1997.

[HHR97b] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. In *Proceedings of the 24th International Colloquium on Automata, Languages, and Programming*, pages 214–224. Springer-Verlag *Lecture Notes in Computer Science #1256*, July 1997.

[HHR97c] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Raising NP lower bounds to parallel NP lower bounds. *SIGACT News*, 28(2):2–13, June 1997.

[HHW97] L. Hemaspaandra, H. Hempel, and G. Wechsung. Self-specifying machines. Technical Report Technical Report TR-654, University of Rochester, Department of Computer Science, Rochester, NY, April 1997.

[HILL91] J. Hastad, R. Impagliazzo, L. Levin, and M. Luby. Construction of a pseudo-random generator from any one-way function. Technical Report 91-068, ICSI, Berkeley, 1991.

[HIS85] J. Hartmanis, N. Immerman, and V. Sewelson. Sparse sets in NP−P: EXPTIME versus NEXPTIME. *Information and Control*, 65(2/3):159–181, 1985.

[HJ95] L. Hemaspaandra and S. Jha. Defying upward and downward separation. *Information and Computation*, 121:1–13, 1995.

[HJRW98] L. Hemaspaandra, Z. Jiang, J. Rothe, and O. Watanabe. Boolean operations, joins, and the extended low hierarchy. *Theoretical Computer Science*, 205(1/2):317–327, 1998.

[HM83] S. Homer and W. Maass. Oracle dependent properties of the lattice of NP sets. *Theoretical Computer Science*, 24(3):279–289, 1983.

[HR] L. Hemaspaandra and J. Rothe. A second step towards complexity-theoretic analogs of Rice's Theorem. *Theoretical Computer Science*. To appear. Available as: University of Rochester Department of Computer Science Technical Report TR-97-662, July 1997.

[HR90] L. Hemachandra and S. Rudich. On the complexity of ranking. *Journal of Computer and System Sciences*, 41(2):251–271, 1990.

[HR92] L. Hemachandra and R. Rubinstein. Separating complexity classes with tally oracles. *Theoretical Computer Science*, 92(2):309–318, 1992.

[HR94] M. Halldorsson and J. Radhakrishnan. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 439–448. ACM Press, 1994.

[HR95] L. Hemaspaandra and J. Rothe. Intersection suffices for boolean hierarchy equivalence. In *Proceedings of the First Annual International Computing and Combinatorics Conference*, pages 430–435. Springer-Verlag *Lecture Notes in Computer Science #959*, August 1995.

[HR97a] E. Hemaspaandra and J. Rothe. Recognizing when greed can approximate maximum independent sets is complete for parallel access to NP. Technical Report Math/Inf/97/14, Friedrich-Schiller-Universität Jena, Jena, Germany, May 1997.

[HR97b] L. Hemaspaandra and J. Rothe. Unambiguous computation: Boolean hierarchies and sparse Turing-complete sets. *SIAM Journal on Computing*, 26(3):634–653, 1997.

[HR98a] E. Hemaspaandra and J. Rothe. Recognizing when greed can approximate maximum independent sets is complete for parallel access to NP. *Information Processing Letters*, 65(3):151–156, February 1998.

[HR98b] L. Hemaspaandra and J. Rothe. A second step towards circuit complexity-theoretic analogs of Rice's Theorem. In *Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science*, pages 418–426. Springer-Verlag *Lecture Notes in Computer Science #1450*, August 1998.

[HR99] L. Hemaspaandra and J. Rothe. Creating strong, total, commutative, associative one-way functions from any one-way function in complexity theory. *Journal of Computer and System Sciences*, 58:648–659, 1999.

[HRW97a] L. Hemaspaandra, J. Rothe, and G. Wechsung. Easy sets and hard certificate schemes. *Acta Informatica*, 34(11):859–879, 1997.

[HRW97b] L. Hemaspaandra, J. Rothe, and G. Wechsung. On sets with easy certificates and the existence of one-way permutations. In *Proceedings of the Third Italian Conference on Algorithms and Complexity*, pages 264–275. Springer-Verlag *Lecture Notes in Computer Science #1203*, March 1997.

[HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[HV95] L. Hemaspaandra and H. Vollmer. The Satanic notations: Counting classes beyond #P and other definitional adventures. *SIGACT News*, 26(1):2–13, 1995.

[HW91] L. Hemachandra and G. Wechsung. Kolmogorov characterizations of complexity classes. *Theoretical Computer Science*, 83:313–322, 1991.

[HW97] E. Hemaspaandra and G. Wechsung. The minimization problem for boolean formulas. In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, pages 575–584. IEEE Computer Society Press, November 1997.

[HY84] J. Hartmanis and Y. Yesha. Computation times of NP sets of different densities. *Theoretical Computer Science*, 34(1/2):17–32, 1984.

[HZ93] L. Hemaspaandra and M. Zimand. Strong forms of balanced immunity. Technical Report Technical Report TR-480, University of Rochester, Department of Computer Science, Rochester, NY, December 1993. Revised, May 1994.

[HZ96] L. Hemaspaandra and M. Zimand. Strong self-reducibility precludes strong immunity. *Mathematical Systems Theory*, 29(5):535–548, 1996.

[IN88] R. Impagliazzo and M. Naor. Decision trees and downward closures. In *Proceedings of the 3rd Structure in Complexity Theory Conference*, pages 29–38. IEEE Computer Society Press, June 1988.

[IR89] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st ACM Symposium on Theory of Computing*, pages 44–61. ACM Press, 1989.

[Joh74] D. Johnson. Worst case behavior of graph coloring algorithms. In *Proceedings of the 5th Southeastern Conference on Combinatorics, Graph Theory, and Computing*, pages 513–527, February/March 1974.

[Kad88] J. Kadin. The polynomial time hierarchy collapses if the boolean hierarchy collapses. *SIAM Journal on Computing*, 17(6):1263–1282, 1988. Erratum appears in the same journal, 20(2):404.

[Kar72] R. Karp. Reducibilities among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103, 1972.

[Kar94] J. Kari. Rice's Theorem for the limit sets of cellular automata. *Theoretical Computer Science*, 127(2):229–254, 1994.

[KL80] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th ACM Symposium on Theory of Computing*, pages 302–309, April 1980. An extended version has also appeared as: Turing machines that take advice, *L'Enseignement Mathématique*, 2nd series 28, 1982, pages 191–209.

[Kle52] S. Kleene. *Introduction to Metamathematics*. D. van Nostrand Company, Inc., New York and Toronto, 1952.

[KMRS97] S. Kurtz, S. Mahaney, J. Royer, and J. Simon. Biological computing. In L. Hemaspaandra and A. Selman, editors, *Complexity Theory Retrospective II*, pages 179–195. Springer-Verlag, 1997.

[Ko85] K. Ko. On some natural complete operators. *Theoretical Computer Science*, 37(1):1–30, 1985.

[Ko89] K. Ko. Relativized polynomial time hierarchies having exactly k levels. *SIAM Journal on Computing*, 18(2):392–408, 1989.

[Ko90] K. Ko. A note on separating the relativized polynomial time hierarchy by immune sets. *R.A.I.R.O. Theoretical Informatics and Applications*, 24(3):229–240, 1990.

[Kol65] A. Kolmogorov. Three approaches for defining the concept of information quantity. *Prob. Inform. Trans.*, 1:1–7, 1965.

[KS85] K. Ko and U. Schöning. On circuit-size complexity and the low hierarchy in NP. *SIAM Journal on Computing*, 14(1):41–51, 1985.

[KST89] J. Köbler, U. Schöning, and J. Torán. On counting and approximation. *Acta Informatica*, 26(4):363–379, 1989.

[KSTT92] J. Köbler, U. Schöning, S. Toda, and J. Torán. Turing machines with few accepting computations and low sets for PP. *Journal of Computer and System Sciences*, 44(2):272–286, 1992.

[KSW87] J. Köbler, U. Schöning, and K. Wagner. The difference and truth-table hierarchies for NP. *R.A.I.R.O. Informatique théorique et Applications*, 21:419–435, 1987.

[Kur83] S. Kurtz. A relativized failure of the Berman-Hartmanis conjecture. Technical Report TR83-001, University of Chicago Department of Computer Science, Chicago, IL, 1983.

[Lad75] R. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22(1):155–171, 1975.

[Lau83] C. Lautemann. BPP and the polynomial hierarchy. *Information Processing Letters*, 17(4):215–217, 1983.

[Lev73] L. Levin. Universal sorting problems. *Problems of Information Transmission*, 9:265–266, 1973.

[Lev84] L. Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61:15–37, 1984.

[Lis] G. Lischke. Towards the actual relationship between NP and exponential time. *Mathematical Logic Quarterly*. To appear. A preliminary version has appeared as: Impossibilities and possibilities of weak separation between NP and exponential time. In *Proceedings of the 5th Structure in Complexity Theory Conference*, pages 245–253. IEEE Computer Society Press, 1990.

[LLS75] R. Ladner, N. Lynch, and A. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1(2):103–124, 1975.

[Lon85] T. Long. On restricting the size of oracles compared with restricting access to oracles. *SIAM Journal on Computing*, 14(3):585–597, 1985. Erratum appears in the same journal, 17(3):628.

[LS86] T. Long and A. Selman. Relativizing complexity classes with sparse oracles. *Journal of the ACM*, 33(3):618–627, 1986.

[LV90] M. Li and P. Vitányi. Applications of Kolmogorov complexity in the theory of computation. In A. Selman, editor, *Complexity Theory Retrospective*, pages 147–203. Springer-Verlag, 1990.

[LV93] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer-Verlag, 1993.

[Mah82] S. Mahaney. Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis. *Journal of Computer and System Sciences*, 25(2):130–143, 1982.

[MMI72] D. Matula, G. Marble, and J. Isaacson. Graph coloring algorithms. In R. Read, editor, *Graph Theory and Computing*, pages 109–122. Academic Press, 1972.

[MS72] A. Meyer and L. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proceedings of the 13th IEEE Symposium on Switching and Automata Theory*, pages 125–129, 1972.

[MU95] I. McLean and A. Urken. *Classics of Social Choice*. University of Michigan Press, Ann Arbor, Michigan, 1995.

[NW94] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.

[OH93] M. Ogiwara and L. Hemachandra. A complexity theory for feasible closure properties. *Journal of Computer and System Sciences*, 46(3):295–325, 1993.

[Pap94] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[PY84] C. Papadimitriou and M. Yannakakis. The complexity of facets (and some facets of complexity). *Journal of Computer and System Sciences*, 28(2):244–259, 1984.

[PZ83] C. Papadimitriou and S. Zachos. Two remarks on the power of counting. In *Proceedings of the 6th GI Conference on Theoretical Computer Science*, pages 269–276. Springer-Verlag *Lecture Notes in Computer Science #145*, 1983.

[Rac82] C. Rackoff. Relativized questions involving probabilistic algorithms. *Journal of the ACM*, 29(1):261–268, 1982.

[Raz87] A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mat. Zametki*, 41(4):598–607, 1987. In Russian. English Translation in *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.

[Reg88] K. Regan. The topology of provability in complexity theory. *Journal of Computer and System Sciences*, 36(3):384–432, 1988.

[Reg96] K. Regan. Index sets and presentations of complexity classes. *Theoretical Computer Science*, 161(1–2):263–287, July 1996.

[RH96] J. Rothe and L. Hemaspaandra. Characterizations of the existence of partial and total one-way permutations. Technical Report Math/Inf/96/7, Friedrich-Schiller-Universität Jena, Institut für Informatik, Jena, Germany, April 1996.

[Ric53] H. Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, 74:358–366, 1953.

[Ric56] H. Rice. On completely recursively enumerable classes and their key arrays. *Journal of Symbolic Logic*, 21:304–341, 1956.

[Rog67] H. Rogers, Jr. *The Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967.

[Rot] J. Rothe. Immunity and simplicity for exact counting and other counting classes. *R.A.I.R.O. Theoretical Informatics and Applications*. To appear. Available as: University of Rochester Department of Computer Science Technical Report TR-98-679, January 1998.

[Rot93] J. Rothe. Some closure properties of GAP-definable classes. Technical Report TR Math/93/6, Friedrich-Schiller-Universität Jena, Jena, Germany, 1993. Appeared as part of: A promise class at least as hard as the polynomial hierarchy. *Journal of Computing and Information*, 1(1):92–107, 1995.

[Rot98a] J. Rothe. Heuristics versus completeness for graph coloring. Technical Report Math/Inf/98/29, Friedrich-Schiller-Universität Jena, Institut für Informatik, Jena, Germany, November 1998.

[Rot98b] J. Rothe. Immunity and simplicity for exact counting and other counting classes. In *Proceedings of the 9th International Conference on Computing and Information*, pages 279–286, June 1998. The conference proceedings are to appear as a special issue of the *Journal of Computing and Information*.

[Rot98c] J. Rothe. Immunity and simplicity for exact counting and other counting classes. Technical Report TR 679, University of Rochester, Rochester, NY, January 1998.

[RR95] K. Regan and J. Royer. On closure properties of bounded two-sided error complexity classes. *Mathematical Systems Theory*, 28(3):229–243, 1995.

[RRW94] R. Rao, J. Rothe, and O. Watanabe. Upward separation for FewP and related classes. *Information Processing Letters*, 52:175–180, 1994.

[RS93] M. Rabi and A. Sherman. Associative one-way functions: A new paradigm for secret-key agreement and digital signatures. Technical Report CS-TR-3183/UMIACS-TR-93-124, Department of Computer Science, University of Maryland, College Park, Maryland, 1993. Available on-line at http://www.cs.umbc.edu/pub/REPORTS/cs-93-18.ps.

[RS97] M. Rabi and A. Sherman. An observation on associative one-way functions in complexity theory. *Information Processing Letters*, 64(2):239–244, 1997.

[Rub86] R. Rubinstein. A note on sets with small generalized Kolmogorov complexity. Technical Report Technical Report TR86-4, Iowa State University, Ames, Iowa, March 1986.

[SB84] U. Schöning and R. Book. Immunity, relativization, and nondeterminism. *SIAM Journal on Computing*, 13(2):329–337, 1984.

[Sch83] U. Schöning. A low and a high hierarchy within NP. *Journal of Computer and System Sciences*, 27:14–28, 1983.

[Sch87] U. Schöning. Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences*, 37:312–323, 1987.

[Sch90] U. Schöning. The power of counting. In A. Selman, editor, *Complexity Theory Retrospective*, pages 204–223. Springer-Verlag, 1990.

[Sel88] A. Selman. Natural self-reducible sets. *SIAM Journal on Computing*, 17(5):989–996, 1988.

[Sel92] A. Selman. A survey of one-way functions in complexity theory. *Mathematical Systems Theory*, 25(3):203–221, 1992.

[Sel95] A. Selman, May 1995. Personal Communication.

[Sim75] J. Simon. *On Some Central Problems in Computational Complexity*. PhD thesis, Cornell University, Ithaca, NY, January 1975. Available as Cornell Department of Computer Science Technical Report TR75-224.

[Sip83a] M. Sipser. Borel sets and circuit complexity. In *Proceedings of the 15th ACM Symposium on Theory of Computing*, pages 61–69, 1983.

[Sip83b] M. Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th ACM Symposium on Theory of Computing*, pages 330–335, 1983.

[Smo87] R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the 19th ACM Symposium on Theory of Computing*, pages 77–82. ACM Press, May 1987.

[Sto73] L. Stockmeyer. Planar 3-colorability is NP-complete. *SIGACT News*, 5(3):19–25, 1973.

[Sto77] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1977.

[TO92] S. Toda and M. Ogiwara. Counting classes are at least as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 21(2):316–328, 1992.

[Tod91a] S. Toda. *Computational Complexity of Counting Complexity Classes*. PhD thesis, Tokyo Institute of Technology, Department of Computer Science, Tokyo, Japan, 1991.

[Tod91b] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[Tor86] L. Torenvliet. *Structural Concepts in Relativised Hierarchies*. PhD thesis, Universiteit van Amsterdam, Amsterdam, The Netherlands, 1986.

[Tor88] J. Torán. *Structural Properties of the Counting Hierarchies*. PhD thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, 1988.

[Tor91] J. Torán. Complexity classes defined by counting quantifiers. *Journal of the ACM*, 38(3):753–774, 1991.

[Tra84] B. Trakhtenbrot. A survey of Russian approaches to *perebor* (brute-force search) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984.

[TT96] D. Tankus and M. Tarsi. Well-covered claw-free graphs. *Journal of Combinatorial Theory*, 66:293–302, 1996.

[Tur36] A. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, ser. 2*, 42:230–265, 1936. Correction, *ibid*, vol. 43, pp. 544–546, 1937.

[TvEB89] L. Torenvliet and P. van Emde Boas. Simplicity, immunity, relativizations and nondeterminism. *Information and Computation*, 80(1):1–17, 1989.

[Val76] L. Valiant. The relative complexity of checking and evaluating. *Information Processing Letters*, 5(1):20–23, 1976.

[Val79a] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

[Val79b] L. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[Vol94] H. Vollmer. *Komplexitätsklassen von Funktionen*. PhD thesis, Universität Würzburg, Institut für Informatik, Würzburg, Germany, 1994. In German.

[VV86] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.

[VW93] H. Vollmer and K. Wagner. The complexity of finding middle elements. *International Journal of Foundations of Computer Science*, 4:293–307, 1993.

[VW95] H. Vollmer and K. Wagner. Complexity classes of optimization functions. *Information and Computation*, 120:198–219, 1995.

[Wag86] K. Wagner. The complexity of combinatorial problems with succinct input representations. *Acta Informatica*, 23:325–356, 1986.

[Wag87] K. Wagner. More complicated questions about maxima and minima, and some closures of NP. *Theoretical Computer Science*, 51:53–80, 1987.

[Wag90] K. Wagner. Bounded query classes. *SIAM Journal on Computing*, 19(5):833–846, 1990.

[Wat88] O. Watanabe. On hardness of one-way functions. *Information Processing Letters*, 27:151–157, 1988.

[Wat92] O. Watanabe. On polynomial-time one-truth-table reducibility to a sparse set. *Journal of Computer and System Sciences*, 44(3):500–516, 1992.

[Wel93] D. Welsh. *Complexity: Knots, Colourings and Counting*. Cambridge University Press, 1993.

[Woo69] D. Wood. A technique for coloring a graph applicable to large scale timetabling problems. *The Computer Journal*, 12:317–319, 1969.

[Wra77] C. Wrathall. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science*, 3:23–33, 1977.

[WW86] K. Wagner and G. Wechsung. *Computational Complexity*. D. Reidel Publishing Company, 1986. Distributors for the U.S.A. and Canada: Kluwer Academic Publishers.

[Yao82] A. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.

[Yao85] A. Yao. Separating the polynomial-time hierarchy by oracles. In *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science*, pages 1–10. IEEE Computer Society Press, 1985.

Ehrenwörtliche Erklärung

Ich erkläre hiermit, daß mir die Habilitationsordnung der Friedrich-Schiller-Universität Jena bekannt ist. Ferner erkläre ich, daß ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quellen gekennzeichnet.

Andere Personen waren an der inhaltlich-materiellen Erstellung der Arbeit nicht beteiligt. Insbesondere habe ich hierfür nicht die entgeltliche Hilfe von Vermittlungs- bzw. Beratungsdiensten in Anspruch genommen. Niemand hat von mir unmittelbar oder mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Arbeit stehen.

Die Arbeit wurde bisher weder im In- noch Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt.

Ich versichere, daß ich nach bestem Wissen die reine Wahrheit gesagt und nichts verschwiegen habe.

Jena, 2. 2. 1999 Jörg-Matthias Rothe

Lebenslauf

	Rothe, Jörg-Matthias
1.11.1966	geboren in Erfurt, Thüringen verheiratet
1973–1981	Besuch der Oberschule in Erfurt
1981–1985	Besuch einer Spezialklasse mathematisch-physikalischer Richtung an der Erweiterten Oberschule in Ilmenau, Thüringen
1985	Abitur
1985–1986	Tätigkeit als Pflegerische Hilfskraft an der Bezirksnervenklinik Hildburghausen
1986–1991	Studium der Mathematik an der Friedrich-Schiller-Universität Jena
16.9.1991	Diplom in Mathematik
9/1991–10/1991	Wissenschaftlicher Mitarbeiter an der Mathematischen Fakultät der Friedrich-Schiller-Universität Jena
11/1991–5/1992	Erhalt eines Graduiertenstipendiums der Friedrich-Schiller-Universität Jena
5/1992–5/1996	Wissenschaftlicher Mitarbeiter am Institut für Informatik der Friedrich-Schiller-Universität Jena
9/1993–6/1994	Forschungsaufenthalt an der University of Rochester in den USA mit einem Jahresstipendium des DAAD
12.10.1995	Promotion, On Some Promise Classes in Structural Complexity Theory, *summa cum laude*, Friedrich-Schiller-Universität Jena
seit 6/1996	Wissenschaftlicher Assistent am Institut für Informatik der Friedrich-Schiller-Universität Jena
9/1997–7/1998	Forschungsaufenthalt an der University of Rochester in den USA mit einem Postdoktoranden-Stipendium des DAAD
seit 10/1998	Erziehungsurlaub

Jena, 2. 2. 1999 Jörg-Matthias Rothe